一百多个经典实例，深入阐释Android应用开发精髓



朱元涛◎编著

Android 应用开发范例大全

- 实录550分钟、165个高清学习视频。
- 详解165个经典实例，每个实例都可以独立解决一类问题。
- 所有实例均取自实际项目开发，既启发思维，又快速提升实战水平。
- 教授精髓，精讲精炼。赠送源码，拿来就用。



超值赠送 DVD

- 15个Android综合项目开发案例
- 35个Android应用开发学习视频

清华大学出版社

Android 应用开发范例大全

朱元涛　编著

清华大学出版社
北　京

内 容 简 介

Android 系统从诞生到现在，在短短的几年时间里，便凭借操作易用性和开发的简洁性，赢得了广大用户和开发者的支持。截至 2014 年 9 月 30 日，Android 系统的市场占有率高达 85%。本书采用实例教学的方式，以 165 个经典应用范例的实现过程，详细讲解了开发各类 Android 应用程序的方法和技巧。

本书共有 14 章，从 UI 界面布局实战开始讲起，依次讲解基本控件应用，事件处理实战，界面显示实战，自动化服务应用实战，文件操作和数据存储实战，电话和短信实战，二维/三维图形、渲染和动画实战，网络实战应用，视频和音频实战应用，手机游戏应用，移动 Web 应用，Google API 服务，传感器实战应用等内容。每一个范例的讲解，都遵循理论联系实际的讲解方式，并详细讲解实例必备的理论知识。

本书几乎涵盖了所有 Android 应用项目开发的主要内容，适合 Android 应用开发者、Android 初/中级读者、Android 爱好者、Android 传感器开发人员、Android 智能家居开发人员、Android 可穿戴设备开发人员的学习，也可以作为相关培训学校和大专院校相关专业的教学用书。

图书在版编目（CIP）数据

Android 应用开发范例大全/朱元涛编著．—北京：清华大学出版社，2015
ISBN 978-7-302-40282-4

I．①A…　II．①朱…　III．①移动终端-应用程序-程序设计-手册　IV．①TN929.53-62

中国版本图书馆 CIP 数据核字（2015）第 106388 号

责任编辑：朱英彪
封面设计：刘　超
版式设计：牛瑞瑞
责任校对：赵丽杰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：203mm×260mm　**印　　张：**33.5　**字　　数：**938 千字
（附 DVD 光盘 1 张）
版　　次：2015 年 7 月第 1 版　**印　　次：**2015 年 7 月第 1 次印刷
印　　数：1～3000
定　　价：86.00 元

产品编号：061879-01

前　　言

2007 年 11 月 5 日，谷歌公司宣布基于 Linux 平台的开源手机操作系统 Android 诞生，该平台号称是首个为移动终端打造的真正开放和完整的系统，本书将带领广大读者领略这款系统的神奇之处。

市场占有率高居第一

截至 2014 年 9 月，Android 在手机市场上的占有率从 2013 年的 68.8%上升到 85%。而 iOS 则从 2013 年的 19.4%下降到 15.5%，WP 系统从原来的 2.7%小幅上升到 3.6%。从数据上看，Android 平台占据了市场的主导地位。

由数据可知，Android 市场的占有率增加幅度较大，WP 市场小幅增长，但 iOS 却有所下降。就目前来看，智能手机的市场已经饱和，大多数用户都在各个平台中转换。而就在这样一个市场上，Android 还增长了 10%左右的占有率实属不易。

为开发人员提供了平台

（1）保证开发人员可以迅速转型为 Android 应用开发

Android 应用程序是通过 Java 语言开发的，只要具备 Java 开发基础，就能很快上手并掌握。作为单独的 Android 应用开发，对 Java 编程门槛的要求并不高，即使没有编程经验，也可以在突击学习 Java 之后学习 Android。另外，Android 完全支持 2D、3D 和数据库，并且和浏览器实现了集成。所以通过 Android 平台，程序员可以迅速、高效地开发出绚丽多彩的应用，如常见的工具、管理和游戏等。

（2）定期举行奖金丰厚的 Android 大赛

为了吸引更多的用户使用 Android 开发，谷歌已经成功举办了奖金为数千万美元的开发者竞赛，鼓励开发人员创建出创意十足且实用的软件。这种大赛对于开发人员来说，不但能提高自己的开发水平，并且高额的奖金也是参赛的动力。

（3）开发人员可以利用自己的作品赚钱

为了能让 Android 平台吸引更多的关注，谷歌提供了一个专门下载 Android 应用的门店——Android Market，网址是 https://play.google.com/store。该门店允许开发人员发布应用程序，也允许 Android 用户下载自己喜欢的程序。作为开发者，需要申请开发者账号，申请后才能将自己的程序上传到 Android Market，并且可以对自己的软件进行定价。只要你的软件程序足够吸引人，就可以获得金钱回报。这样实现了程序员学习和赚钱两不误，所以吸引了更多开发人员加入到 Android 大军中来。

本书的内容

本书采用实例教学的方式，通过 165 个经典应用范例的实现过程，详细讲解了开发各类 Android

应用程序的方法和技巧。本书共有 14 章，从 UI 界面布局实战开始讲起，依次讲解了基本控件应用，事件处理实战，界面显示实战，自动化服务应用实战，文件操作和数据存储实战，电话和短信实战，二维/三维图形、渲染和动画实战，网络实战应用，视频和音频实战应用，手机游戏应用，移动 Web 应用，Google API 服务，传感器实战应用等内容。

本书的版本

Android 系统自 2008 年 9 月发布第一个版本 1.1 以来，截至 2014 年 10 月发布最新版本 5.0，一共存在十多个版本。由此可见，Android 系统升级频率较高，一年中最少有两个新版本诞生。如果过于追求新版本，会造成力不从心的结果。所以在此建议广大读者不必追求最新的版本，只需关注最流行的版本即可。据官方统计，截至 2014 年 10 月 25 日，占据前 3 位的版本分别是 Android 4.3、Android 4.4 和 Android 4.2，其实这 3 个版本的区别并不是很大，只是在某领域的细节上进行了更新。为了及时体验 Android 系统的最新功能，本书使用的版本是主流的 Android 5.0。

本书特色

本书内容十分丰富，讲解细致，目的是通过一本图书，提供多本图书的价值，读者可以根据自己的需要有选择地阅读。在内容的编写上，本书具有以下特色：

（1）内容全面，讲解细致

本书几乎涵盖了开发 Android 应用所涉及的所有领域，详细讲解了每一个典型应用项目的实现过程，每一个范例都力求用翔实易懂的语言展现在读者面前。

（2）理论和实践相结合

为了使广大读者彻底理解 Android 应用项目开发的各个知识点，在讲解每一个范例时，都详细剖析了对应的必备理论知识，确保读者能够真正明白该范例的原理。

（3）章节独立，自由阅读

本书的每一章内容都可以独自成书，读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行针对性的学习。阅读本书会带来很大的快乐。

（4）实例典型，实用性强

本书讲解现实中最典型应用实例的实现方法和架构技巧，这些外设应用都是在商业项目中最需要的构成部分。读者可以直接将本书中的知识应用到自己的项目中，实现无缝对接。

（5）附配资源丰富

本书配有丰富的学习资源，除源代码、PPT 之外，还实录了 165 个高清学习视频，每个实例都可以独立解决某一类问题，快速提高实战水平。除此以外，本书额外赠送了 35 个 Android 应用开发学习视频，以及 15 个 Android 应用开发综合案例，包括仿小米录音机、音乐播放器、跟踪定位系统、仿陌陌交友系统、手势音乐播放器、智能家居系统、湿度测试仪、象棋游戏、抢滩登陆游戏、九宫格数独游戏、健康饮食系统、仓库管理系统、个人财务系统、仿去哪儿酒店预定系统、仿开心网客户端等。通过这些附配资源，读者的学习过程会更加方便、快捷。

读者对象

本书适合 Android 应用开发者、Android 初/中级读者、Android 爱好者、Android 传感器开发人员、Android 智能家居开发人员、Android 可穿戴设备开发人员学习，也可以作为相关培训学校和大专院校相关专业的教学用书。

参与本书编写的人员还有周秀、付松柏、邓才兵、钟世礼、谭贞军、张加春、王教明、万春潮、郭慧玲、侯恩静、程娟、王文忠、陈强、何子夜、李天祥、周锐、朱桂英、张元亮、张韶青、秦丹枫。

本书在编写过程中，得到了清华大学出版社的大力支持，在此表示由衷的感谢。另外也十分感谢我的家人，在我写作时给予了巨大支持。因编者水平有限，错误和不尽如人意之处在所难免，恳请读者提出意见或建议，以便修订并使之更臻完善。另外，我们提供了售后支持网站（http://www.chubanbook.com/）和 QQ 群（192153124），读者朋友如有疑问可以在此提出，一定会得到满意的答复。

编　者

目　录

第 1 章　UI 界面布局实战

对于网站开发人员来说，网站结构和界面设计是影响浏览用户第一视觉的关键。而对于 Android 应用开发来说，除了功能强大的应用程序外，屏幕界面效果也是影响程序质量的重要元素，因为消费者喜欢的是界面美观且功能强大的软件产品。在设计优美界面之前，一定要先对屏幕进行布局。在本章的内容中，将以具体实例来介绍布局 Android 屏幕的知识，为读者学习本书后面的知识打下基础。

1.1　第一个 Android 应用程序

实例 001	第一个 Android 应用程序
源码路径	光盘:\daima\001
视频路径	光盘:\视频\001
实例必备	001. 搭建 Android 应用开发环境.pdf ① 安装 Android SDK 的系统要求 ② 安装 JDK ③ 获取并安装 Eclipse 和 Android SDK ④ 安装 ADT ⑤ 设定 Android SDK Home ⑥ 验证开发环境 ⑦ 创建 Android 虚拟设备（AVD） ⑧ 启动 AVD 模拟器

本实例的功能是在手机屏幕中显示问候语“你好我的朋友!”，在开始之前先做一个简单的流程规划，如图 1-1 所示。

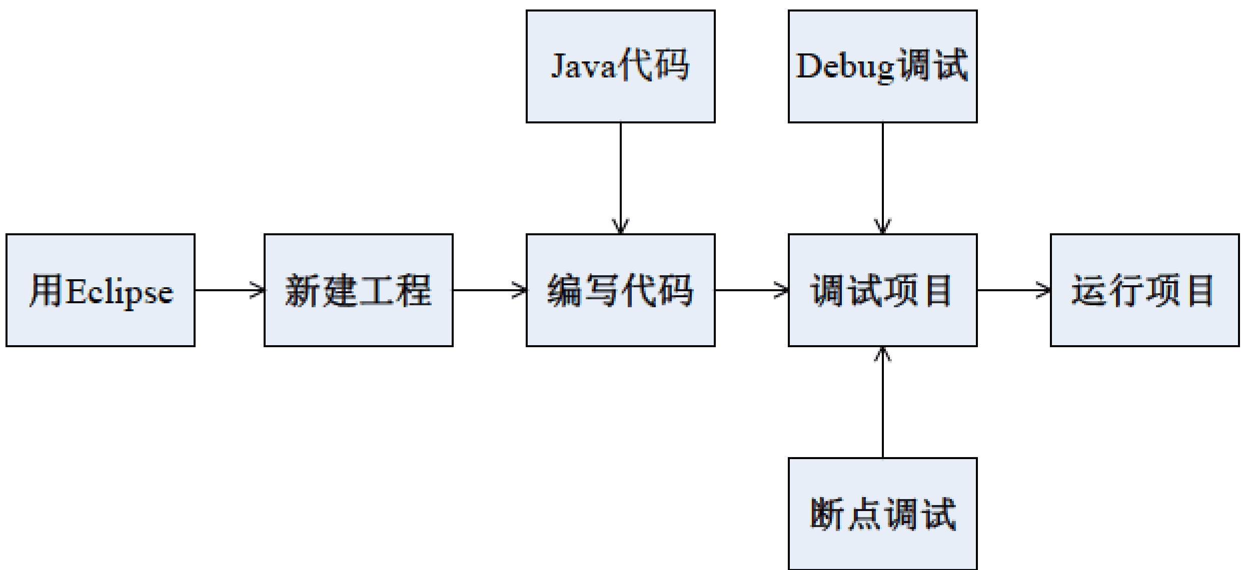


图 1-1　规划流程图

在接下来的内容中，将详细讲解本实例的具体实现流程。

1.1.1 使用 Eclipse 新建 Android 工程

（1）打开 Eclipse，依次选择 File | New | Project 命令新建一个工程，如图 1-2 所示。

（2）选择 Android Project 选项，单击 Next 按钮。

（3）在弹出的 New Android Application 对话框中设置工程信息，如图 1-3 所示。

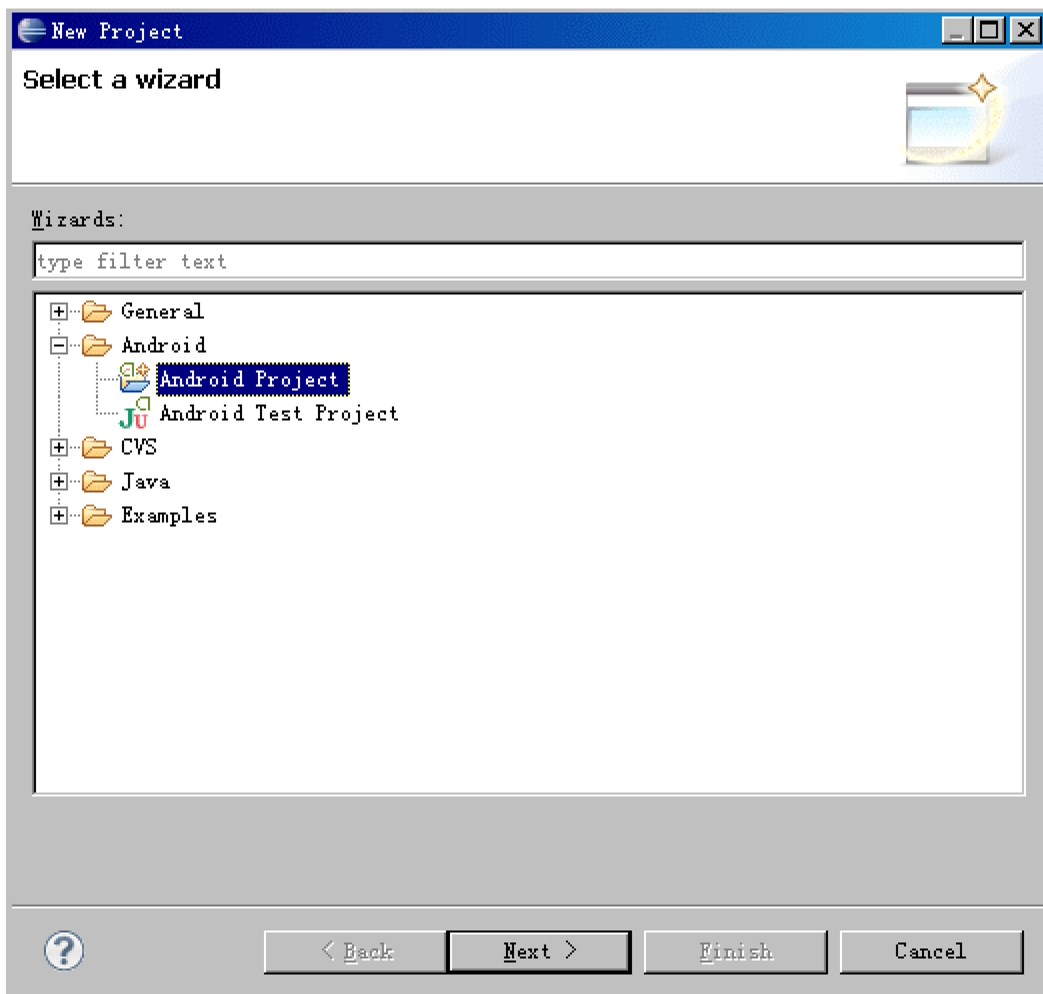


图 1-2 新建工程文件

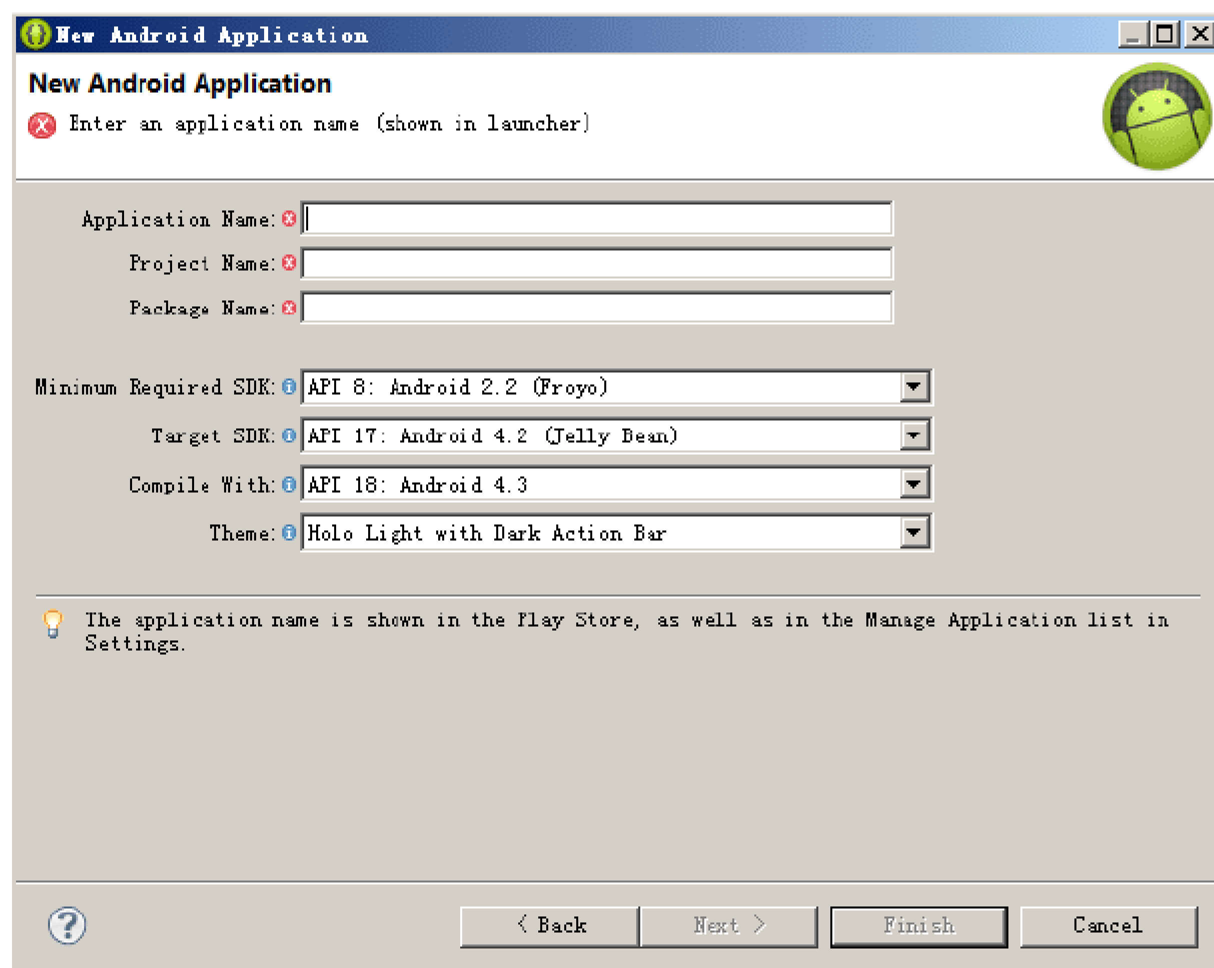


图 1-3 设置工程

在图 1-3 所示的界面中依次设置工程名字、包名字、Activity 名字和应用名字。

1.1.2 编写代码和代码分析

现在已经创建了一个名为 first 的工程文件，现在打开文件 first.java，会显示自动生成的如下代码。

```
package first.a;
import android.app.Activity;
import android.os.Bundle;
public class fistMM extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

如果此时运行程序，将不会显示任何内容。此时可以稍微修改上述代码，让程序输出“你好我的朋友!”，具体代码如下所示。

```
package first.a;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.widget.TextView;

public class fistMM extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText("你好我的朋友！ ");
        setContentView(tv);
    }
}
```

经过修改后，可以在屏幕中输出“你好我的朋友!”，完全符合预期的要求。

1.1.3　调试程序

Android 调试一般分为 3 个步骤，分别是设置断点、Debug 调试和断点调试。

（1）设置断点。此处的设置断点和 Java 中的方法一样，可以通过双击代码左边的区域进行断点设置，如图 1-4 所示。

为了调试方便，可以设置显示代码的行数。只需在代码左侧的空白部分右击，在弹出的快捷菜单中选择 Show Line Numbers 命令，如图 1-5 所示。

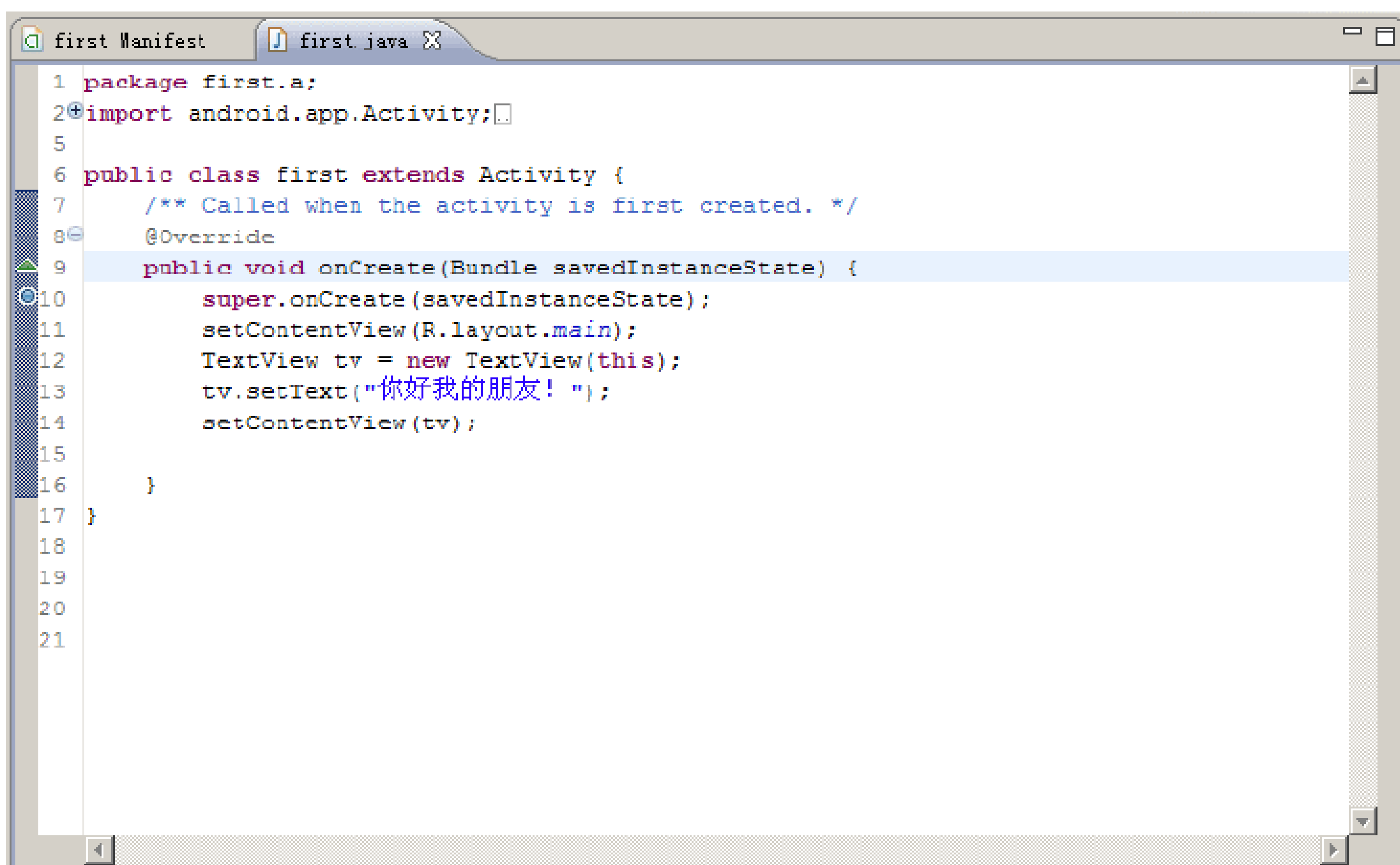


图 1-4　设置断点

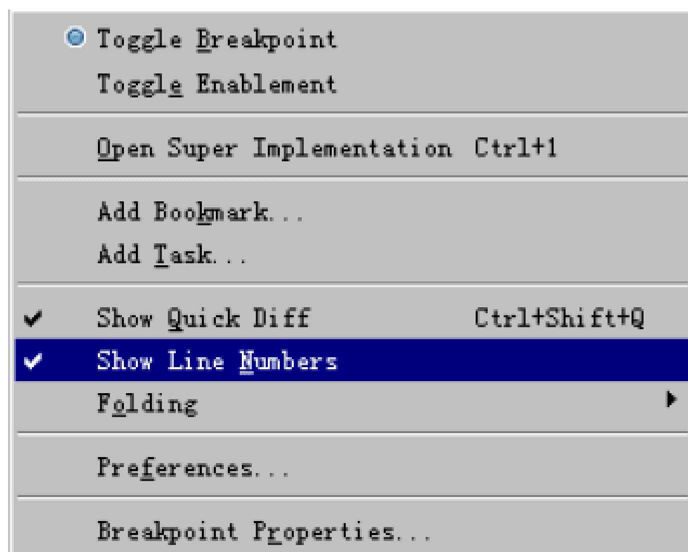


图 1-5　显示行数

（2）Debug 调试。Debug Android 调试项目的方法和普通 Debug Java 调试项目的方法类似，唯一不同的是在选择调试项目时选择 Android Application 命令。具体方法是右击项目名，在弹出的快捷菜单中选择 Debug As | Android Application 命令，如图 1-6 所示。

（3）断点调试。可以进行单步调试，具体调试方法和调试普通 Java 程序的方法类似，调试界面如图 1-7 所示。

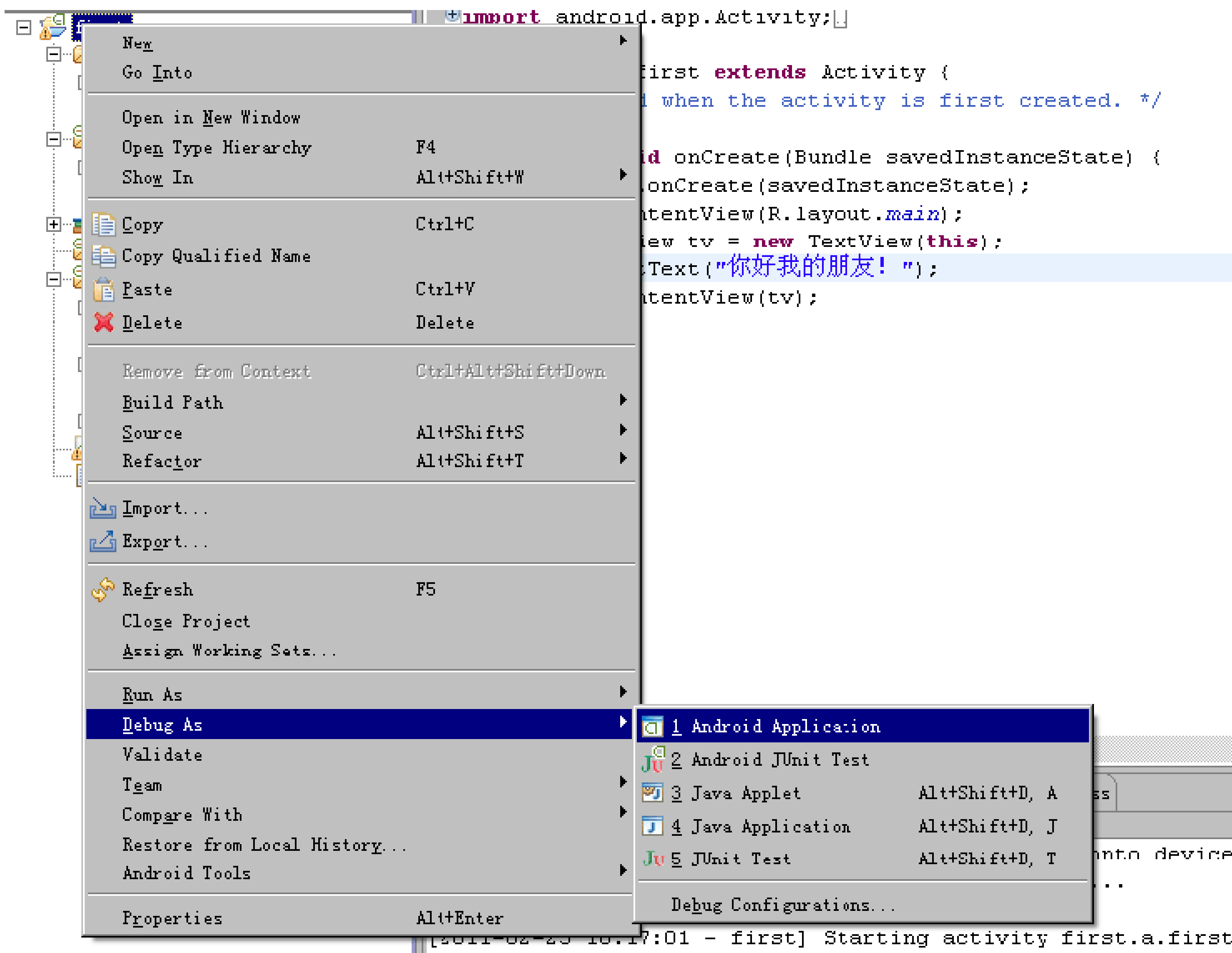


图 1-6　Debug 调试项目

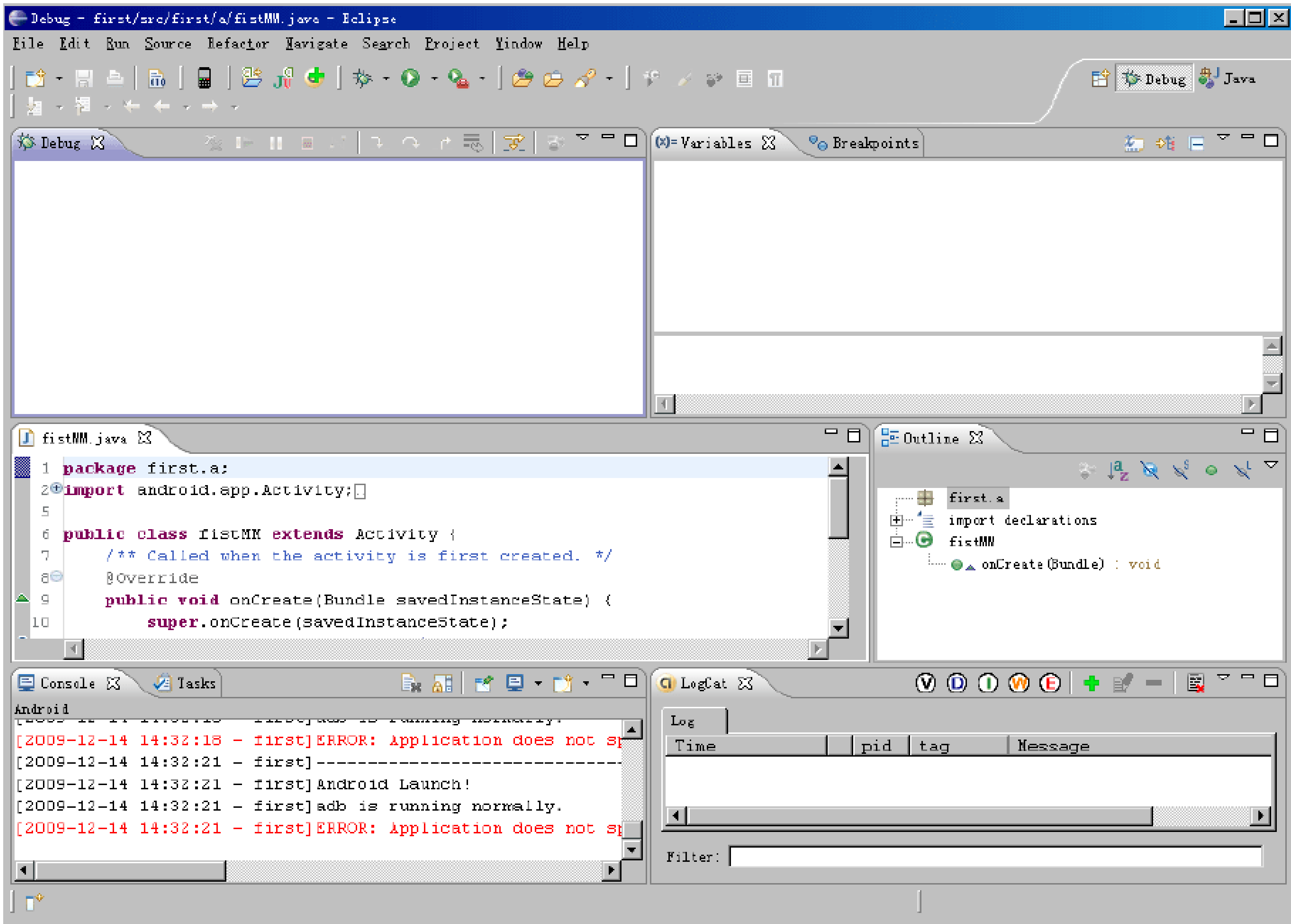


图 1-7　调试界面

1.1.4　运行项目

将上述代码保存后就可运行这段程序了，具体过程如下所示。

（1）右击项目名，在弹出的快捷菜单中选择 Run As | Android Application，如图 1-8 所示。

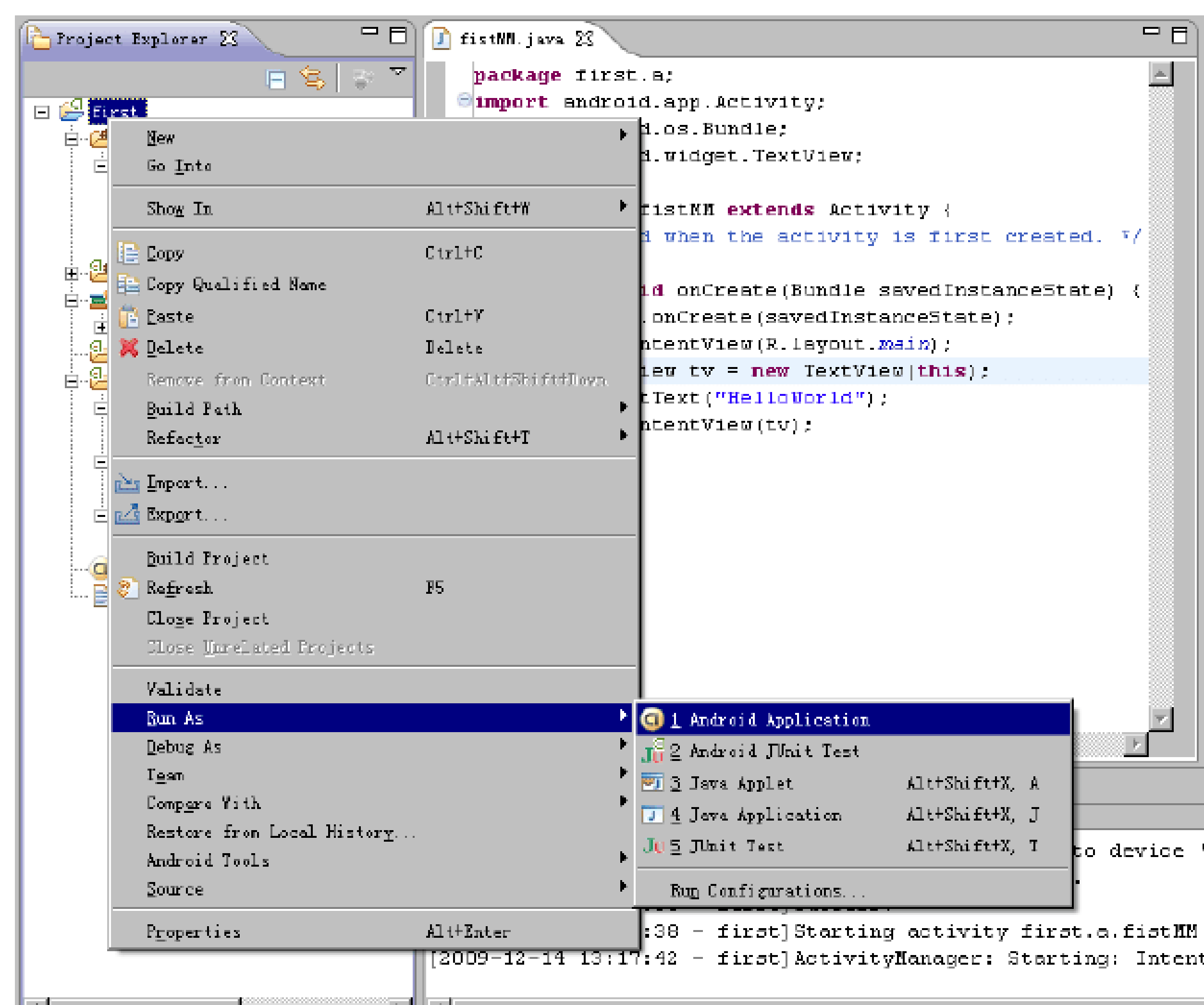


图 1-8　开始运行

（2）此时工程开始运行，运行完成后在屏幕中输出“你好我的朋友！”，如图 1-9 所示。

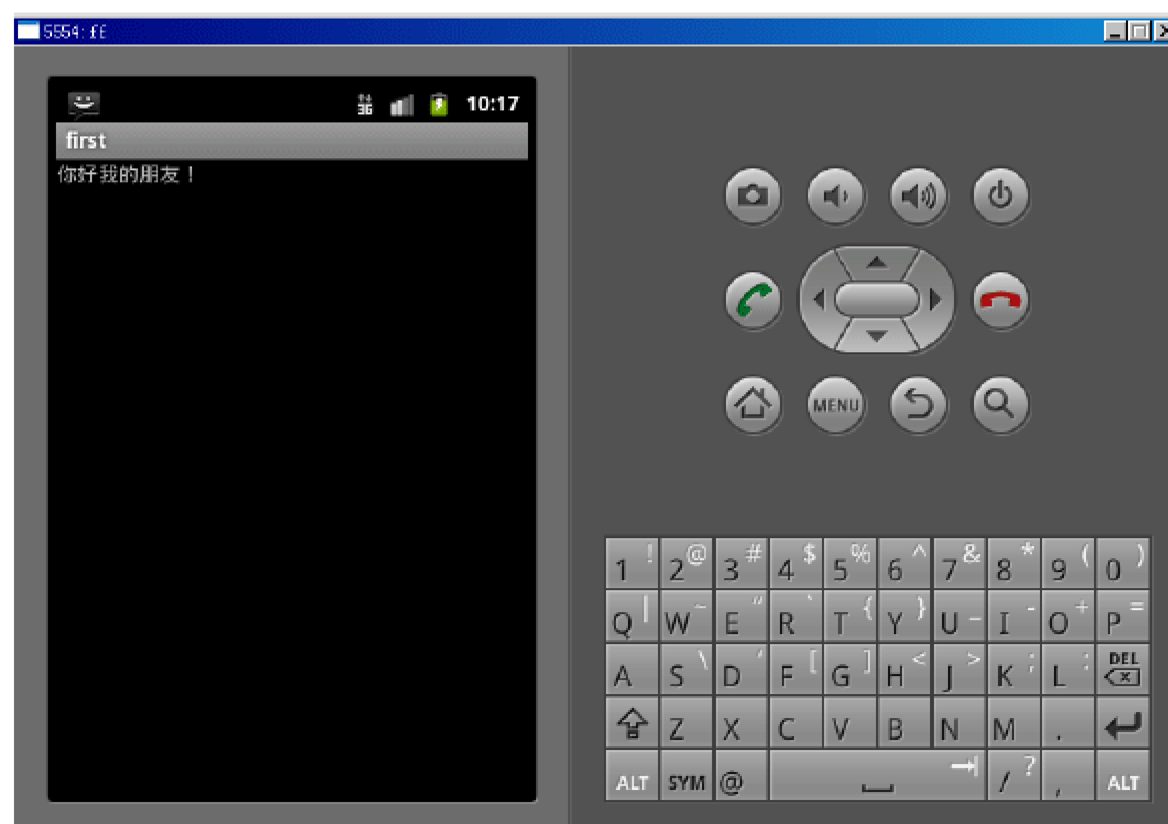


图 1-9　运行结果

1.2　使用线性布局（LinearLayout）

实例 002	使用线性布局（LinearLayout）
源码路径	光盘:\daima\002
视频路径	光盘:\视频\002
实例必备	002. View 视图组件.pdf ① View 的常用属性和方法 ② Viewgroup 容器 ③ ViewManager 类

1.2.1 实例说明

一个 Android 程序是由一个或多个 Activity 组成的，Activity 是一个 UI 容器，本身不在用户界面中显示出来。其中类 View 起了一个重要的作用，View 是一个最基本的 UI 类，几乎所有的 UI 组件都是继承于 View 而实现的。View 的使用格式如下：

```
android.view.View
```

其主要功能如下：

☑ 为指定的屏幕矩形区域存储布局和内容。

☑ 处理尺寸和布局，绘制，焦点改变，翻屏，按键，手势。

☑ widget 基类。

线性布局即 LinearLayout 布局，是 Android 屏幕中常用的布局方式，是一个 ViewGroup 以线性方向显示其子视图（view）元素，即垂直的或水平的。

1.2.2 具体实现

编写布局文件 res\layout\main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:orientation="horizontal">
    <Button android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello, I am a Button1"
                android:layout_weight="1"
                />
    <Button android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button2"
    android:layout_weight="1"
    />
    <Button android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button3"
    android:layout_weight="1"
    />
    <Button android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button4"
    android:layout_weight="1"
    />
    <Button android:id="@+id/button5"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, I am a Button5"
    android:layout_weight="1"
    />
</LinearLayout>
```

在上述代码中，在根 LinearLayout 视图组（ViewGroup）中包含了 5 个 Button，其子元素是以线性方式（horizontal，水平的）布局，运行效果如图 1-10 所示。

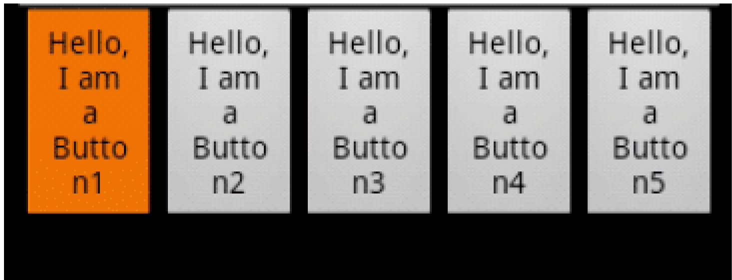


图 1-10　LinearLayout 布局

1.3　使用相对布局（RelativeLayout）

实例 003	使用相对布局（RelativeLayout）
源码路径	光盘:\daima\003
视频路径	光盘:\视频\003
实例必备	003.Android UI 布局的方式.pdf ① 使用 XML 布局 ② 在 Java 代码中控制布局

1.3.1　实例说明

相对布局（RelativeLayout）是指一个 ViewGroup 以相对位置显示其子视图（view）元素，一个视图可以指定相对于它的兄弟视图的位置（例如在给定视图的左边或者下面）或相对于 RelativeLayout 的特定区域的位置（例如底部对齐，或中间偏左）。相对布局是设计用户界面的有力工具，因为它消除了嵌套视图组。如果发现使用了多个嵌套的 LinearLayout 视图组，则可以考虑使用一个 RelativeLayout 视图组。本实例中演示了在屏幕中使用相对布局的方法。

1.3.2　具体实现

编写布局文件 res\layout\main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel" />
</RelativeLayout>
```

执行后的效果如图 1-11 所示。

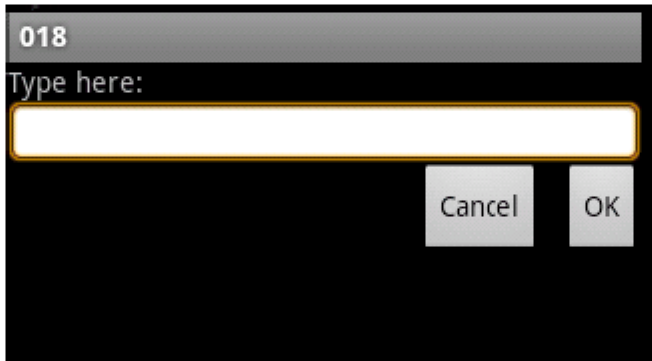


图 1-11　执行效果

1.4　使用表格布局（TableLayout）

实例 004	使用表格布局（TableLayout）
源码路径	光盘:\daima\004
视频路径	光盘:\视频\004
实例必备	004.Android 布局管理器详解.pdf ① 布局组件 Layout 的语法格式 ② 5 个最为常用的 Layout 实现类

1.4.1　实例说明

表格布局（TableLayout）是一个 ViewGroup 以表格显示其子视图（view）元素，即行和列标识一

个视图的位置。其实 Android 的表格布局与 HTML 中的表格布局非常类似，TableRow 就像 HTML 表格的<tr>标记。表格布局通常用于把子元素放入行与列中，不显示行、列或是单元格边界线，但是单元格不能横跨行，如 HTML 中一样。效果如图 1-12 所示。

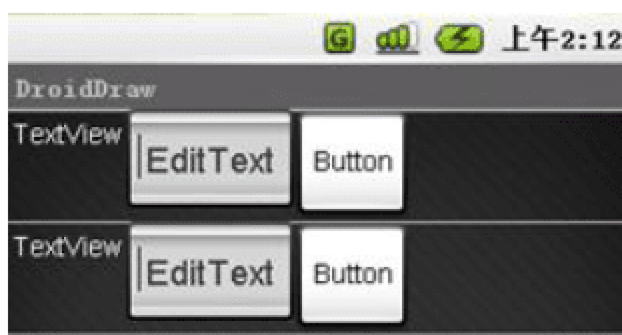


图 1-12　表格布局

在使用表格布局时需要注意以下 3 点。

- ☑ android:shrinkColumns：对应的方法是 setShrinkAllColumns (boolean)，作用是设置表格的列是否收缩（列编号从 0 开始，下同），如果有多列则用逗号隔开（下同），如 android:shrinkColumns="0,1,2"，即表格的第 1、2、3 列的内容是收缩的，以适合屏幕，不会挤出屏幕。
- ☑ android:collapseColumns：对应的方法是 setColumnCollapsed(int,boolean)，作用是设置表格的列是否隐藏。
- ☑ android:stretchColumns：对应的方法是 setStretchAllColumns(boolean)，作用是设置表格的列是否拉伸。

1.4.2　具体实现

编写布局文件 res\layout\main.xml，具体实现代码如下所示。

```
<TableRow>
  <Button android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button1"
            android:layout_column="0"
            />
   <Button android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button2"
            android:layout_column="1"
            />
</TableRow>
<TableRow>
  <Button android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button3"
            android:layout_column="1"
            />
  <Button android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button4"
            android:layout_column="1"
            />
</TableRow>
<TableRow>
```

```
        <Button android:id="@+id/button5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello, I am a Button5"
                android:layout_column="2"
                />
    </TableRow>
</TableLayout>
```

执行后的效果如图 1-13 所示。

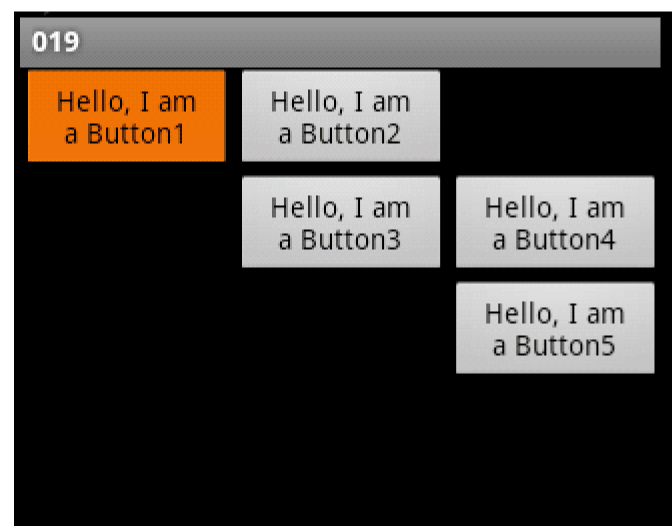


图 1-13　执行效果

1.5　使用绝对布局（AbsoluteLayout）

实例 005	使用绝对布局（AbsoluteLayout）
源码路径	光盘:\daima\005
视频路径	光盘:\视频\005
实例必备	005.绝对布局 AbsoluteLayout.pdf ① 结构 ② 公共方法 ③ 受保护方法

1.5.1　实例说明

绝对布局（AbsoluteLayout）是一个 ViewGroup 以绝对方式显示其子视图（view）元素，即以坐标的方式来定位在屏幕上的位置。这种布局方式很好理解，在布局文件或编程中设置 view 的坐标，从而绝对地定位。

1.5.2　具体实现

编写布局文件 res\layout\main.xml，具体实现代码如下所示。

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/AbsoluteLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

```
    <TextView android:id="@+id/txtIntro"
      android:text="绝对布局"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_x="20dip"
      android:layout_y="20dip">
    </TextView>
</AbsoluteLayout>
```

执行后的效果如图 1-14 所示。

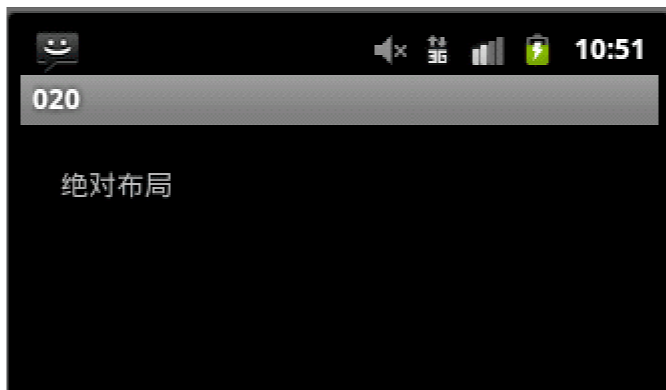


图 1-14　执行效果

1.6　使用标签布局（TabLayout）

实例 006	使用标签布局（TabLayout）
源码路径	光盘:\daima\006
视频路径	光盘:\视频\006
实例必备	006.线性布局 LinearLayout.pdf ① LinearLayout 常用属性 ② LinearLayout 子元素控制

1.6.1　实例说明

标签布局（TabLayout）是一个 ViewGroup 以标签的方式显示其子视图（view）元素，就像在 Firefox 中的一个窗口中显示多个网页一样。为了创建一个标签 UI（Tabbed UI），需要使用到 TabHost 和 TabWidget。TabHost 必须是布局的根节点，包含为了显示标签的 TabWidget 和显示标签内容的 FrameLayout。

1.6.2　具体实现

布局文件 res\layout\main.xml 的具体实现代码如下所示。

```
<TableRow>
    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
     android:text="7"
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text="8"
     />
```

```
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text="9"
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" / "
     />
</TableRow>

<TableRow>
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 4 "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 5"
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 6 "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" * "
     />
</TableRow>

<TableRow>
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 1 "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 2 "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 3 "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" . "
     />
</TableRow>

<TableRow>
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" 0 "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" = "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" - "
     />
    <Button android:layout_width="wrap_content"
     android:layout_height="wrap_content" android:text=" + "
     />
```

```
</TableRow>

</TableLayout>
```

执行后将显示计算器的效果，如图 1-15 所示。

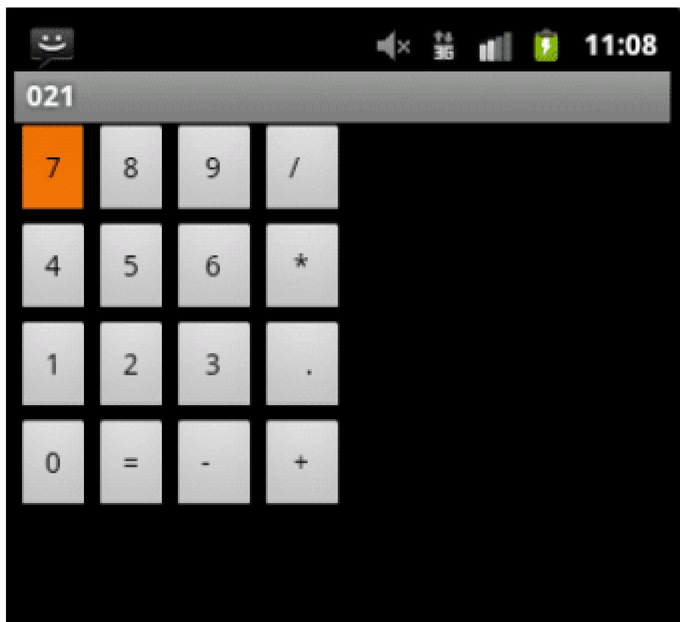


图 1-15　计算器效果

在上述实例中，使用了按钮来实现计算器的按键效果，其实为了提高项目的美观性，可以使用素材图片来实现案件效果。在实现标签内容时，可使用标签在同一个活动中交换视图，在完全隔离的活动之间改变。根据需要，选择不同的方式，但是如果每个标签提供不同的用户活动，应为每个标签选择隔离的活动，因此可以更好地以分离的组管理应用程序，而不是一个巨大的应用程序和布局。

1.7　使用层布局（FrameLayout）

实例 007	使用层布局（FrameLayout）
源码路径	光盘:\daima\007
视频路径	光盘:\视频\007
实例必备	007.层布局 FrameLayout.pdf

1.7.1　实例说明

层布局（FrameLayout）是最简单的一个布局对象，被定制为屏幕上的一个空白备用区域，之后可以在其中填充一个单一对象，例如一张要发布的图片。所有的子元素将会固定在屏幕的左上角；用户不能为 FrameLayout 中的一个子元素指定一个位置。后一个子元素将会直接在前一个子元素之上进行覆盖填充，把前一个子元素部分或全部挡住（除非后一个子元素是透明的）。

1.7.2　具体实现

编写布局文件 res\layout\main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent">
    <TextView
        android:text="big"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="50pt"/>
    <TextView
        android:text="middle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20pt"/>
    <TextView
        android:text="small"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="10pt"/>
</FrameLayout>
```

执行后因为多层重叠而发生重影效果，如图 1-16 所示。

从图 1-16 所示效果可以看出，FrameLayout 能够将组件显示在屏幕的左上角，后面的组件覆盖前面的组件。该布局 container 可以用来占有屏幕的某块区域来显示单一的对象，可以包含多个 widgets 或者 container，但是所有被包含的 widgets 或是 container 必须被固定到屏幕的左上角，并且一层覆盖一层，而不能通过为一个 widgets 或者是 container 指定一个位置。在 container 中所包含的 widgets 或者 container 的队列采用的是堆栈结构，最后添加进来的 widgets 或者 container 显示在最上面。所以后一个 widgets 或 container 将会直接覆盖在前一个 widgets 或 container 之上，把它们部分或全部挡住。除非后一个 widgets 或 container 是透明的，否则必须得到 FrameLayout Container 的允许。



图 1-16　重影效果

1.8　Layout 布局的综合应用

实例 008	Layout 布局的综合应用
源码路径	光盘:\daima\008
视频路径	光盘:\视频\008
实例必备	008.绝对布局 AbsoluteLayout.pdf ① 结构 ② 公共方法 ③ 受保护方法

1.8.1　实例说明

布局就像容器，里面可以放置很多控件。布局里面还可以套用其他的布局，这样就可以实现界面

的多样化以及设计的灵活性。使用布局组件 Layout 的格式如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
```

在一个布局容器中可以包括 0 个或多个布局容器。线性的布局方式，要么使用上下的方式添加控件，要么使用左右的方式添加控件。

其实本章前面的实例都是属于 Layout 布局的范畴，如线性布局（LinearLayout）和相对布局（RelativeLayout）等。

1.8.2　具体实现

文件 ActivityMain.java 是项目的主文件，功能是调用各个公用文件来实现具体的功能，具体说明如下所示。

- ☑ 通过函数 setContentView(R.layout.main)实现了 Activity 和布局文件 main.xml 的关联。
- ☑ 使用对象 button0、button1、button2 和 button3 分别代表 4 个 Button 控件。
- ☑ 为了查看不同布局的效果，分别为各个 Button 控件设置了单击监听器，每一个监听器都会跳转到一个新的 Activity。

（1）文件 ActivityMain.java 的主要代码如下所示。

```
        setContentView(R.layout.main);
        button0 = (Button) findViewById(R.id.button0);
        button0.setOnClickListener(listener0);
        button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(listener1);
        button2 = (Button) findViewById(R.id.button2);
        button1.setOnClickListener(listener2);
        button3 = (Button) findViewById(R.id.button3);
        button3.setOnClickListener(listener3);
```

（2）编写布局文件 main.xml，此段代码是一个典型的 LinearLayout 布局样式，如下所示。

```
    <Button android:id="@+id/button0"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="使用 FrameLayout"/>
    <Button android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="使用 RelativeLayout"/>
    <Button android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="LinearLayout 和 RelativeLayout"/>
    <Button android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="使用 TableLayout"/>
```

（3）编写单击第一个按钮 button0 的处理代码。单击后会在新界面中显示一幅图片，此界面是用 FrameLayout 布局的。在文件 activity_frame_layout.xml 中定义了这幅图片的显示样式，即在 FrameLayout 布局中添加了一个图片显示组件 ImageView 元素。具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout android:id="@+id/left"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"                    /**x 轴方向填充空间*/
    android:layout_height="fill_parent"                   /**y 轴方向填充空间*/
    >
    <ImageView android:id="@+id/photo"                    /**定义组件的 id*/
        android:src="@drawable/bg"
        android:layout_width="wrap_content"               /**宽度能包容图片*/
        android:layout_height="wrap_content"              /**高度能包容图片*/
    />
</FrameLayout>
```

对上述代码的几点说明如下所示。

☑ android:id：通过 android:id 来访问定义的显示组件 ImageView 元素。

☑ android:layout_width="fill_parent"：表示 FrameLayout 布局可以在 x 轴方向填充的空间。

☑ android:layout_height="fill_parent"：表示 FrameLayout 布局可以在 y 轴方向填充的空间。

☑ android:layout_width="wrap_content"和 android:layout_height="wrap_content"：表示 ImageView 只需将图片完全包含即可。

（4）编写单击第二个按钮 button1 的处理代码。单击后会显示一个输入用户名的表单界面效果，此功能是通过 RelativeLayout 实现的。对应的布局文件是 relative_layout.xml。主要实现代码如下所示。

```
    <TextView android:id="@+id/label" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="请输入用户名：" />
    <!--
        该 EditText 放置在上边 id 为 label 的 TextView 的下边
    -->
    <EditText android:id="@+id/entry" android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label" />
    <!--
        "取消"按钮和容器的右边齐平，并且设置左边的边距为 10dip
    -->
    <Button android:id="@+id/cancel" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip" android:text="取消" />
    <!--
        "确定"按钮在"取消"按钮的左侧，并且和"取消"按钮的高度齐平
    -->
    <Button android:id="@+id/ok" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/cancel"
        android:layout_alignTop="@id/cancel" android:text="确定" />
```

对上述代码的主要说明如下所示。

☑ android:id：定义组件的 ID。

☑ android:layout_width：用于设置组件的宽度，主要有如下两种设置方式。

➢ fill_parent：填充父容器。

  - ➢ wrap_content：仅仅包容内容即可。
- ☑ android:layout_height：定义组件的高度。
- ☑ android:layout_below="@id/label"：将此组件放在 ID 为 label 的组件下方。这是一种很经典的布局方式，优点是适配性很强，并且不用关心具体的细节，在不同屏幕、不同手机设备上都是通用的。
- ☑ android:layout_alignParentRight="true"：是相对布局，表示和父容器的右边对齐。
- ☑ android:layout_marginLeft="10dip"：设置 ID 为 cancel 的 Button 的左边距为 10 dip。
- ☑ android:layout_toLeftOf="@id/cancel"：设置此组件在 ID 为 cancel 的组件的左边。
- ☑ android:layout_alignTop="@id/cancel"：设置此组件和 ID 为 cancel 的组件的高度对齐。

（5）编写单击第三个按钮 button2 后的处理代码。单击后会显示一系列文本信息，此功能是通过 LinearLayout 和 RelativeLayout 布局联合实现的。

- ☑ 布局第 a 组第 a 项和第 a 组第 b 项。

在文件 left.xml 中通过 RelativeLayout 布局实现此功能，主要实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/left"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/view1" android:background="@drawable/blue"
        android:layout_width="fill_parent"
        android:layout_height="50px" android:text="第 a 组第 a 项" />
    <TextView android:id="@+id/view2"
        android:background="@drawable/yellow"
        android:layout_width="fill_parent"
        android:layout_height="50px" android:layout_below="@id/view1"
        android:text="第 a 组第 b 项" />
</RelativeLayout>
```

在上述代码中，使用了如下两个高度都是 50 像素的 TextView。

  - ➢ 第一个 TextView：通过"@drawable/blue"设置其背景颜色是 blue。
  - ➢ 第二个 TextView：通过 android:layout_below="@id/view1"设置其位置位于第一个 TextView 的下方。

- ☑ 布局第 b 组第 a 项和第 b 组第 b 项。

在文件 right.xml 中使用另外一个 RelativeLayout 实现此布局功能，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/right"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/right_view1"
        android:background="@drawable/yellow" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="第 b 组第 a 项" />
    <TextView android:id="@+id/right_view2"
        android:background="@drawable/blue"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
            android:layout_below="@id/right_view1" android:text="第 b 组第 b 项" />
</RelativeLayout>
```

注意：上述代码和文件 left.xml 类似，此处不再进行详细介绍。

☑ 实现 Layout 和 Activity 的关联。

在 Activity 中，为了使用方便，可以自行构建一个 Layout。为了实现 Layout 和 Activity 的关联，可以在文件 ActivityLayout.java 中编写如下实现代码。

```
public class ActivityLayout extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /**创建一个 Layout **/
        LinearLayout layoutMain = new LinearLayout(this);
        layoutMain.setOrientation(LinearLayout.HORIZONTAL);
        /**实现 Layout 和 Activity 的关联**/
        setContentView(layoutMain);
        /**得到一个 LayoutInflater 对象，此对象可以对 XML 布局文件进行解析，并生成一个 view**/
        LayoutInflater inflate = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        RelativeLayout layoutLeft = (RelativeLayout) inflate.inflate(
                R.layout.left, null);
        RelativeLayout layoutRight = (RelativeLayout) inflate.inflate(
                R.layout.right, null);
        /**生成一个可以供 Layout 使用的 LayoutParams**/
        RelativeLayout.LayoutParams relParam = new RelativeLayout.LayoutParams(
                RelativeLayout.LayoutParams.WRAP_CONTENT,
                RelativeLayout.LayoutParams.WRAP_CONTENT);
/**将 layoutLeft 添加到 layoutMain，第一个参数是添加进去的 view，第二、三个分别是 view 的高度和宽度**/
layoutMain.addView(layoutLeft, 100, 100);
/**将 layoutRight 添加到 layoutMain，第二个参数是一个 RelativeLayout.LayoutParams**/
layoutMain.addView(layoutRight, relParam);
    }
```

（6）编写单击第四个按钮 button3 后的处理代码。单击后会显示一个整齐排列的表单，此功能是通过 TableLayout 实现的，实现文件 activity_table_layout.xml 的主要代码如下所示。

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView android:text="用户名:" android:textStyle="bold"
            android:gravity="right" android:padding="3dip" />
        <EditText android:id="@+id/username" android:padding="3dip"
            android:scrollHorizontally="true" />
    </TableRow>
    <TableRow>
        <TextView android:text="密码:" android:textStyle="bold"
            android:gravity="right" android:padding="3dip" />
        <EditText android:id="@+id/password" android:password="true"
            android:padding="3dip" android:scrollHorizontally="true" />
    </TableRow>
```

```
        <TableRow android:gravity="right">
            <Button android:id="@+id/cancel"
                android:text="取消" />
            <Button android:id="@+id/login"
                android:text="登录" />
        </TableRow>
</TableLayout>
```

在上述代码中，使用标签 TableLayout 定义了一个表格布局，然后通过 TableRow 标签定义了表格布局里的一行，用户可以根据需要在每一行中加入自己需要的组件。

运行后的初始效果如图 1-17 所示，单击“使用 FrameLayout”按钮后会显示设置的图片，如图 1-18 所示。单击“使用 RelativeLayout”按钮后会显示一个信息输入界面，如图 1-19 所示。单击“使用 LinearLayout 和 RelativeLayout”按钮后会显示 4 块不同样式的区域块，如图 1-20 所示。单击“使用 TableLayout”按钮后会显示一个用户登录表单，如图 1-21 所示。



图 1-17　初始效果



图 1-18　显示图片



图 1-19　显示输入用户名

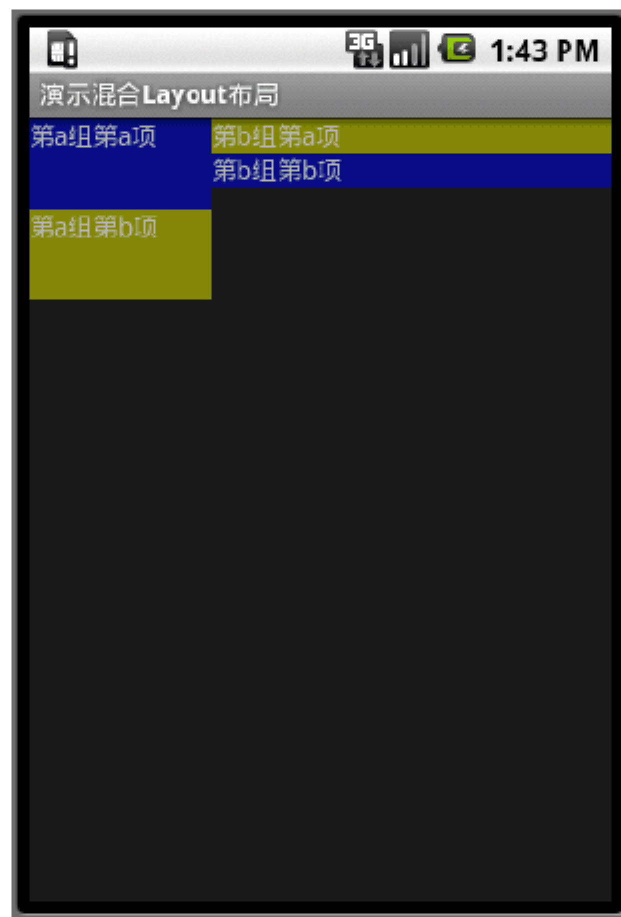


图 1-20　4 块不同样式的区域块



图 1-21　用户登录表单

通过上述实例演示了联合使用 Layout 布局的知识，在具体布局时，读者需要注意几点。例如，在

使用 FrameLayout 布局方式显示一幅图片时，设置的图片不能太大，特别是当前数码相机拍的照片几乎都是数 MB 大小，如果太大，单击“使用 FrameLayout”按钮后不会成功显示设置的图片，如图 1-22 所示。

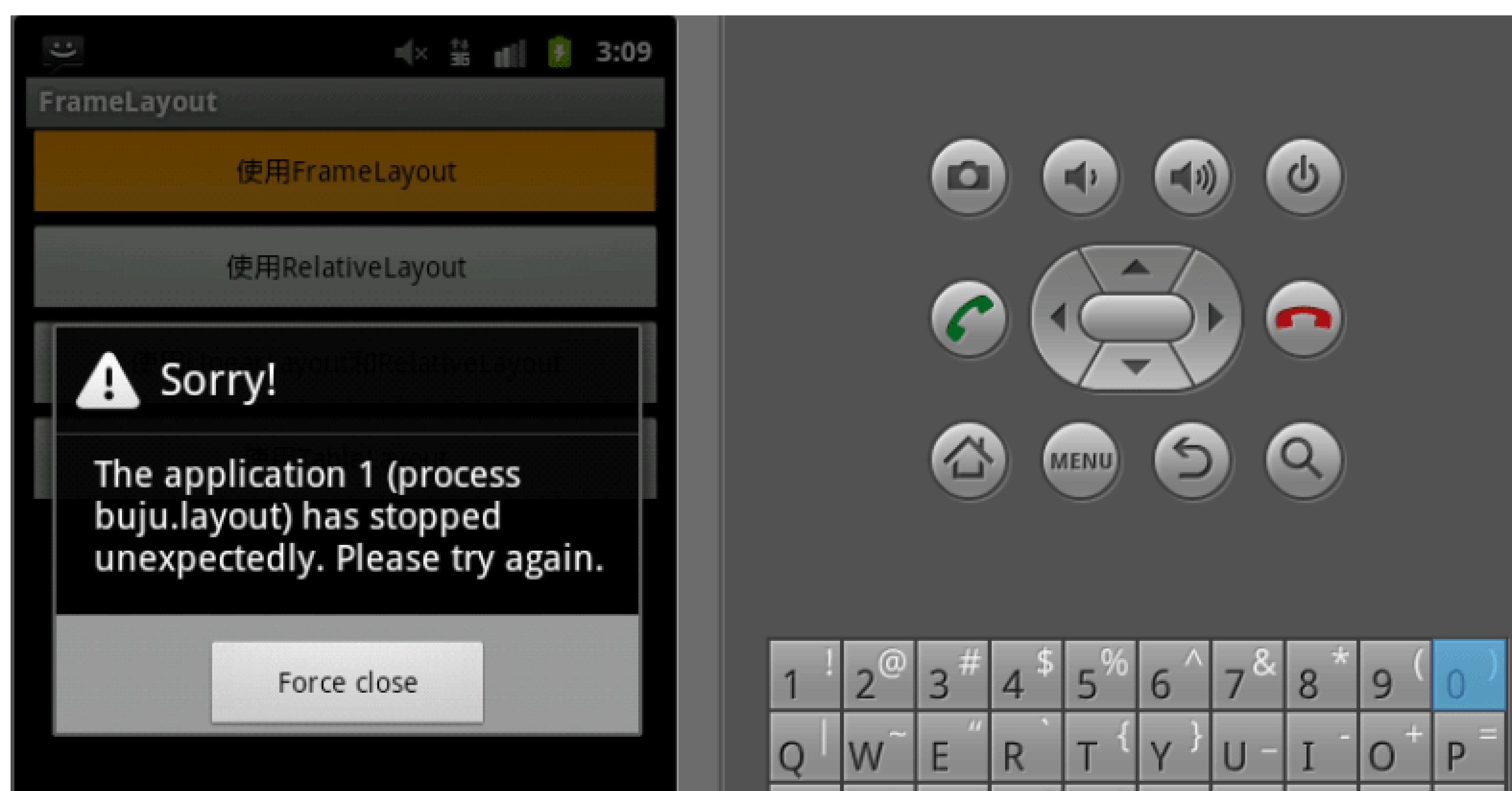


图 1-22　不能显示太大的图片

第 2 章　基本控件应用

对于手机和平板等移动设备来说，几乎所有的应用功能都需要用控件来实现。控件就如同 Web 开发中的一个模块，通过调用这些控件能够实现对应的功能效果。本章将通过具体实例的实现过程，来详细讲解在 Android 系统中使用各个常用控件的基本方法。

2.1　创建一个桌面组件 Widget

实例 009	创建一个桌面组件 Widget
源码路径	光盘:\daima\009\MainActivity.java
视频路径	光盘:\视频\009
实例必备	009.创建一个 Widget 组件.pdf

2.1.1　实例说明

Widget 是一个桌面组件，从名字就可以知道是实现桌面布局的。Widget 这个名称来自一位名叫 Rose 的苹果公司工程师。在 1998 年的一天，Rose 在自己的苹果操作系统桌面玩一个可以更换皮肤的 MP3 播放器时突发奇想：如果在桌面上运行的所有工具都能够更换皮肤或外观，那将是一件很酷的事情，Rose 对此感到十分兴奋，并给这个酷酷的玩意儿起了个名字叫 Konfabulator。

Android 本身已经自带了时钟、音乐播放器、相框和 Google 搜索 4 个 Widget 程序，不过这并不能阻止大家开发更加美观，功能更丰富的版本。另外，微博、RSS 订阅、股市信息、天气预报这些 Widget 也都有流行的可能。

2.1.2　具体实现

（1）使用 Eclipse 创建一个 MainActivity 作为应用程序的入口，自动生成的主文件是 MainActivity.java。

（2）编写布局文件 main.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
</LinearLayout>
```

通过上述代码，在手机屏幕中使用了 Widget 组件。执行后不会显示任何信息，这是因为没有在里

面添加任何元素。由此可见，Widget 组件只是起了一个“容器”的作用，只需把需要显示的屏幕元素添加到这个“容器”里面即可。

注意：本章接下来的实例代码都保存在本实例项目中，即本章剩余实例的源码都保存在“光盘:\daima\009”目录下，这样做的目的是展示在 Widget 组件中“盛装”主要屏幕元素的效果。

2.2　使用 Button 控件实现按钮效果

实例 010	在屏幕中实现一个按钮效果
源码路径	光盘:\daima\009
视频路径	光盘:\视频\010
实例必备	010.使用按钮 Button.pdf

2.2.1　实例说明

在本书前面的示例中，已经使用过 Button 控件，这是一个按钮控件。在日常应用时，当单击 Button 后会触发一个事件，这个事件会实现用户需要的功能。例如，用户输入一些信息，单击“确定”或“取消”按钮后，会实现对应的功能。

2.2.2　具体实现

（1）修改布局文件 main.xml，在其中添加一个 TextView 控件和一个 Button 控件。

（2）在文件 MainActivity.java 中，先通过 findViewByID()获取 TextView 控件和 Button 控件的资源，然后为 Button 控件添加事件监听器 Button.OnClickListener()，最后定义处理事件处理程序。主要代码如下所示。

```
//获取 TextView 文本和 Button 控件的资源
show= (TextView)findViewById(R.id.show_TextView);
press=(Button)findViewById(R.id.Click_Button);
//为 Button 控件添加事件监听器  Button.OnClickListener()
press.setOnClickListener(new Button.OnClickListener(){
                @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                }
            });
//定义处理事件处理程序
press.setOnClickListener(new Button.OnClickListener(){
                @Override
                public void onClick(View v) {
                    //单击按钮后输出一段文本
                    show.setText(“哎呦，button 被点了一下”);
                }
            });
```

运行后首先显示一个“按钮+文本”样式的界面，当单击“这是 button”按钮后会执行单击事件，即执行定义的事件处理程序，如图 2-1 所示。

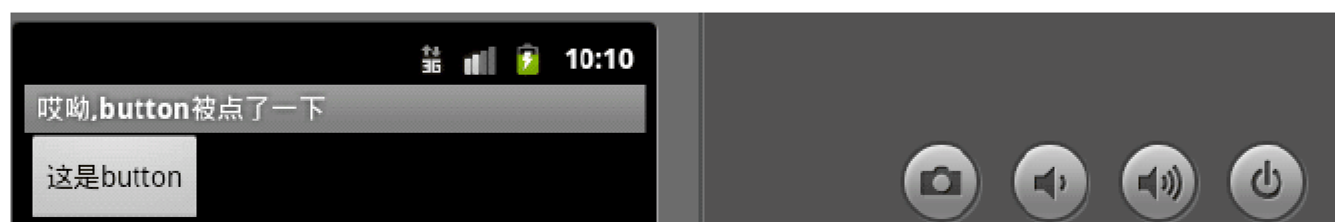


图 2-1　执行效果

2.3　使用 TextView 控件显示文字

实例 011	在屏幕中显示文字
源码路径	光盘:\daima\009
视频路径	光盘:\视频\011
实例必备	011.使用文本框 TextView.pdf ① 使用 TextView ② 使用 TextView 实现颜色变换

2.3.1　实例说明

在手机屏幕中可以通过文本框控件 TextView 来显示文本。在使用 TextView 控件时，通常需要遵循如下 5 个步骤。

（1）导入 TextView 包，具体代码如下所示。

```
import android.widget.TextView;
```

（2）在文件 MainActivity.java 中声明一个 TextView，代码如下所示。

```
private TextView mTextView01;
```

（3）在文件 main.xml 中定义一个 TextView 对象 TextView01，代码如下所示。

```
<TextView android:text="TextView01"
                    android:id="@+id/TextView01"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_x="61px"
                    android:layout_y="69px">
</TextView>
```

（4）利用 findViewById()方法获取 main.xml 中的 TextView，代码如下所示。

```
mTextView01 = (TextView) findViewById(R.id.TextView01);
```

（5）设置 TextView 标签内容，代码如下所示。

```
String str_2 = "欢迎来到 Android 世界...";
mTextView01.setText(str_2);
```

2.3.2　具体实现

修改文件 MainActivity.java，在其中分别添加 12 个 TextView 对象变量、1 个 LinearLayout 对象变

量、1 个 WC 整数变量和 1 个 LinearLayout.LayoutParams 变量，主要实现代码如下所示。

```
/* 定义使用的对象*/
private LinearLayout myLayout;
private LinearLayout.LayoutParams layoutP;
private int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
private TextView black_TV, blue_TV, cyan_TV, dkgray_TV,
                 gray_TV, green_TV,ltgray_TV, magenta_TV, red_TV,
                 transparent_TV, white_TV, yellow_TV;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /*实例化一个 LinearLayout 布局对象*/
    myLayout = new LinearLayout(this);
    /*设置 LinearLayout 的布局为垂直布局*/
    myLayout.setOrientation(LinearLayout.VERTICAL);
    /*设置 LinearLayout 布局背景图片*/
    myLayout.setBackgroundResource(R.drawable.back);
    /*加载主屏布局*/
    setContentView(myLayout);
    /*实例化一个 LinearLayout 布局参数，用来添加 View*/
    layoutP = new LinearLayout.LayoutParams(WC, WC);
    /*构造实例化 TextView 对象*/
    constructTextView();
    /*把 TextView 添加到 LinearLayout 布局中*/
    addTextView();
    /*设置 TextView 文本颜色*/
    setTextViewColor();
    /*设置 TextView 文本内容*/
    setTextViewText();
}
/*设置 TextView 文本内容*/
public void setTextViewText() {
    black_TV.setText("黑色");
    blue_TV.setText("蓝色");
    cyan_TV.setText("青绿色");
    dkgray_TV.setText("灰黑色");
    gray_TV.setText("灰色");
    green_TV.setText("绿色");
    ltgray_TV.setText("浅灰色");
    magenta_TV.setText("红紫色");
    red_TV.setText("红色");
    transparent_TV.setText("透明");
    white_TV.setText("白色");
    yellow_TV.setText("黄色");
}
/*设置 TextView 文本颜色*/
public void setTextViewColor() {
    black_TV.setTextColor(Color.BLACK);
    blue_TV.setTextColor(Color.BLUE);
    dkgray_TV.setTextColor(Color.DKGRAY);
    gray_TV.setTextColor(Color.GRAY);
```

```
        green_TV.setTextColor(Color.GREEN);
        ltgray_TV.setTextColor(Color.LTGRAY);
        magenta_TV.setTextColor(Color.MAGENTA);
        red_TV.setTextColor(Color.RED);
        transparent_TV.setTextColor(Color.TRANSPARENT);
        white_TV.setTextColor(Color.WHITE);
        yellow_TV.setTextColor(Color.YELLOW);
}
/*构造实例化 TextView 对象*/
public void constructTextView() {
        black_TV = new TextView(this);
        blue_TV = new TextView(this);
        cyan_TV = new TextView(this);
        dkgray_TV = new TextView(this);
        gray_TV = new TextView(this);
        green_TV = new TextView(this);
        ltgray_TV = new TextView(this);
        magenta_TV = new TextView(this);
        red_TV = new TextView(this);
        transparent_TV = new TextView(this);
        white_TV = new TextView(this);
        yellow_TV = new TextView(this);
}
/*把 TextView 添加到 LinearLayout 布局中*/
public void addTextView() {
        myLayout.addView(black_TV, layoutP);
        myLayout.addView(blue_TV, layoutP);
        myLayout.addView(cyan_TV, layoutP);
        myLayout.addView(dkgray_TV, layoutP);
        myLayout.addView(gray_TV, layoutP);
        myLayout.addView(green_TV, layoutP);
        myLayout.addView(ltgray_TV, layoutP);
        myLayout.addView(magenta_TV, layoutP);
        myLayout.addView(red_TV, layoutP);
        myLayout.addView(transparent_TV, layoutP);
        myLayout.addView(white_TV, layoutP);
        myLayout.addView(yellow_TV, layoutP);
}
```

执行后的效果如图 2-2 所示。

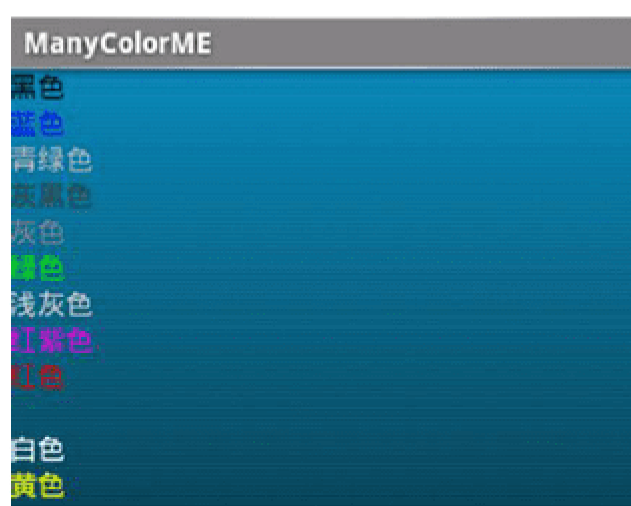


图 2-2　执行效果

由上述实例的实现过程可知，可以设置 TextView 控件的文本颜色。

2.4 设置 TextView 的字体

实例 012	设置手机屏幕中的字体
源码路径	光盘:\daima\009
视频路径	光盘:\视频\012
实例必备	012.在代码中更改 TextView 文字颜色.pdf

2.4.1 实例说明

在 Android 手机系统中，可以使用 TextView 设置屏幕中静态域的字体。在计算机系统中，使用 Typeface 表示字体的风格，具体有如下两种类型。

（1）int Style 类型，具体说明如表 2-1 所示。

表 2-1 int Style 类型说明

字 体	说 明
BOLD	粗体
BOLD_ITALIC	粗斜体
ITALIC	斜体
NORMAL	普通字体

（2）Typeface 类型，具体说明如表 2-2 所示。

表 2-2 Typeface 类型说明

字 体	说 明
DEFAULT	默认字体
DEFAULT_BOLD	默认粗体
MONOSPACE	单间隔字体
SANS_SERIF	无衬线字体
SERIF	衬线字体

2.4.2 具体实现

修改文件 MainActivity.java 以实现多种字体样式的显示，主要实现代码如下所示。

```
public void setTextSizeOf() {
    //设置绘制的文本大小，该值必须大于 0
    bold_TV.setTextSize(24.0f);
    bold_italic_TV.setTextSize(24.0f);
    default_TV.setTextSize(24.0f);
    default_bold_TV.setTextSize(24.0f);
    italic_TV.setTextSize(24.0f);
    monospace_TV.setTextSize(24.0f);
    normal_TV.setTextSize(24.0f);
```

```
        sans_serif_TV.setTextSize(24.0f);
        serif_TV.setTextSize(24.0f);
    }
    public void setTextViewText() {
        /*设置文本*/
        bold_TV.setText("BOLD");
        bold_italic_TV.setText("BOLD_ITALIC");
        default_TV.setText("DEFAULT");
        default_bold_TV.setText("DEFAULT_BOLD");
        italic_TV.setText("ITALIC");
        monospace_TV.setText("MONOSPACE");
        normal_TV.setText("NORMAL");
        sans_serif_TV.setText("SANS_SERIF");
        serif_TV.setText("SERIF");
    }
    public void setStyleOfFont() {
        /*设置字体风格*/
        bold_TV.setTypeface(null, Typeface.BOLD);
        bold_italic_TV.setTypeface(null, Typeface.BOLD_ITALIC);
        default_TV.setTypeface(Typeface.DEFAULT);
        default_bold_TV.setTypeface(Typeface.DEFAULT_BOLD);
        italic_TV.setTypeface(null, Typeface.ITALIC);
        monospace_TV.setTypeface(Typeface.MONOSPACE);
        normal_TV.setTypeface(null, Typeface.NORMAL);
        sans_serif_TV.setTypeface(Typeface.SANS_SERIF);
        serif_TV.setTypeface(Typeface.SERIF);
    }
```

执行后的效果如图 2-3 所示。



图 2-3 执行效果

2.5 使用 EditText 控件显示编辑框

实例 013	在屏幕中显示编辑框
源码路径	光盘:\daima\009
视频路径	光盘:\视频\013
实例必备	013.EditText 控件属性大全.pdf

2.5.1 实例说明

在手机屏幕中，可以像网页一样显示可输入文本信息的文本框，此功能是通过编辑框控件 EditText

实现的。编辑框控件 EditText 的用法和 TextView 类似，能生成一个可编辑的文本框。

2.5.2 具体实现

（1）在主窗口界面中添加一个 EditText 控件，然后设定其监听器在接收到单击事件时，程序打开 EditText 的界面。定义文件 editview.xml 来布局程序打开的 EditText 界面，具体代码如下所示。

```
//供用户输入值
<EditText android:id="@+id/edit_text"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="这里可以输入文字" />
   //用于获取输入的值
      <Button android:id="@+id/get_edit_view_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="获取 EditView 的值" />
```

（2）编写事件处理文件 EditTextActivity.java，主要代码如下所示。

```
private Button.OnClickListener get_edit_view_button_listener = new Button.OnClickListener() {
  /**响应代码，显示 EditText 中的值*/
      public void onClick(View v) {
            EditText edit_text = (EditText) findViewById(R.id.edit_text);
            CharSequence edit_text_value = edit_text.getText();
            setTitle("EditText 的值:"+edit_text_value);
      }
};
```

执行后先显示默认的文本和输入框，如图 2-4 所示；输入一段文本并单击“获取 EditView 的值”按钮后会获取输入的文字，并显示出输入的文字，如图 2-5 所示。



图 2-4 初始效果

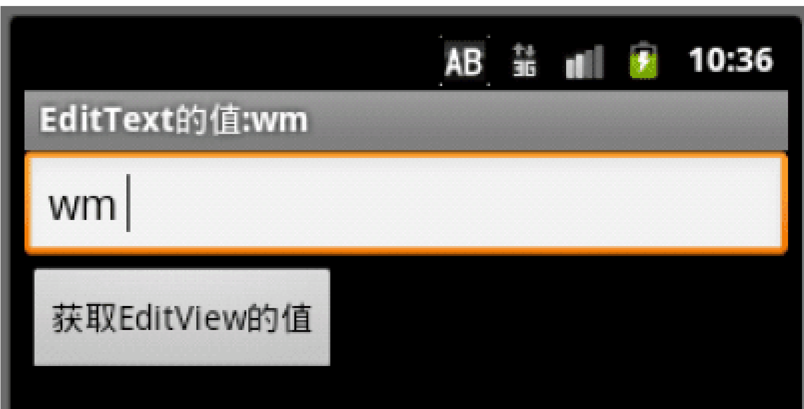


图 2-5 运行效果

2.6 使用 CheckBox 控件显示复选框

实例 014	在屏幕中显示复选框
源码路径	光盘:\daima\009
视频路径	光盘:\视频\014
实例必备	014.CheckBox 控件详解.pdf

2.6.1　实例说明

网页中经常要用到复选框，复选框提供一个制造单一选择开关的方法，包括一个小框和一个标签。典型的复选框有一个小的“X”（或者设置的其他类型）或是空的，这依靠项目是否被选择来决定的。在手机屏幕中也可以实现复选框的效果，此功能是通过 CheckBox 控件实现的。CheckBox 控件能够为用户提供输入信息，用户可以一次性选择多个选项。在 Android 中，使用 CheckBox 控件也需要在 XML 布局文件中定义。

2.6.2　具体实现

（1）编写布局文件 check_box.xml，插入了拥有 4 个可选项的 CheckBox 控件供用户选择，然后插入了一个 Button 控件来响应用户的选择单击事件。

（2）编写单击按钮事件的处理文件 CheckBoxActivity.java，把用户选中的选项值显示在 Title 上。主要代码如下所示。

```
    private void find_and_modify_text_view() {
        plain_cb = (CheckBox) findViewById(R.id.plain_cb);
        serif_cb = (CheckBox) findViewById(R.id.serif_cb);
        italic_cb = (CheckBox) findViewById(R.id.italic_cb);
        bold_cb = (CheckBox) findViewById(R.id.bold_cb);
        Button get_view_button = (Button) findViewById(R.id.get_view_button);
        get_view_button.setOnClickListener(get_view_button_listener);
    }

    private Button.OnClickListener get_view_button_listener = new Button.OnClickListener() {
        public void onClick(View v) {
            String r = "";
            if (plain_cb.isChecked()) {
                r = r + "," + plain_cb.getText();
            }
            if (serif_cb.isChecked()) {
                r = r + "," + serif_cb.getText();
            }
            if (italic_cb.isChecked()) {
                r = r + "," + italic_cb.getText();
            }
            if (bold_cb.isChecked()) {
                r = r + "," + bold_cb.getText();
            }
            setTitle("Checked: " + r);
        }
    };
}
```

执行后先显示 4 个选项值供用户选择，如图 2-6 所示。用户选择某些选项并单击“获取复选框的值”按钮，将在屏幕顶端用文本提示已经选中的复选框信息，如图 2-7 所示。



图 2-6　初始效果



图 2-7　运行效果

2.7　使用 RadioGroup 控件显示单选按钮

实例 015	在屏幕中显示单选按钮
源码路径	光盘:\daima\009
视频路径	光盘:\视频\015
实例必备	015.控件 CheckBox/RadioGroup/RadioButton 的常用属性.pdf

2.7.1　实例说明

无论是上一个实例讲解的 CheckBox（复选框）控件，还是在本实例将要讲解的 RadioButton（单选按钮）控件，这两个控件都只有选中和未选中状态。不同的是，RadioButton 是单选按钮，需要放置到一个 RadioGroup 中，同一时刻一个 RadioGroup 中只能有一个按钮处于选中状态。控件 CheckBox 和 RadioButton 常用方法及说明如表 2-3 所示。

表 2-3　常用方法说明

属　性　名	说　　明
isChecked()	判断是否被选中，选中则返回 True，否则返回 False
performClick()	调用 ClickListener 监听器，即模拟依次单击操作
setChecked()	通过传入的参数设置控件状态
Toggle()	置反控件的当前状态
setOnCheckedChangeListener()	为控件设置 OnCheckedChangeListener 监听器

与复选框相比，单选按钮只能选中一项，是图形用户界面上的一种控件。单选按钮允许用户在一组选项中选择其中的一个选项，其外观一般是一个空白的圆圈，而在圆圈旁边则通常有一个文字的标签。其用途除了描述之外，还可用于选中该选项：当用户单击标签，所对应的选择钮就会被选中。已选中的单选按钮一般会在圆圈内加上一小圆点。单项选择控件 RadioGroup 是和多项选择控件 CheckBox 相对应的，区别就是只能供用户选择一个选项。

2.7.2　具体实现

（1）编写布局文件 radio_group.xml，在其中设置 4 个选项供用户选择。

（2）编写事件处理文件 RadioGroupActivity.java，主要代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.radio_group);
      setTitle("RadioGroupActivity");
      mRadioGroup = (RadioGroup) findViewById(R.id.menu);
      Button clearButton = (Button) findViewById(R.id.clear);
      clearButton.setOnClickListener(this);
}
```

当用户单击"清除"按钮后会使用 setTitle 修改 Title 值为 RadioGroupActivity，然后获取 RadioGroup 对象和按钮对象。执行后先显示 4 个选项值供用户选择，如图 2-8 所示；用户选中一个选项并单击"清除"按钮后会清除选择的选项，如图 2-9 所示。



图 2-8　初始效果

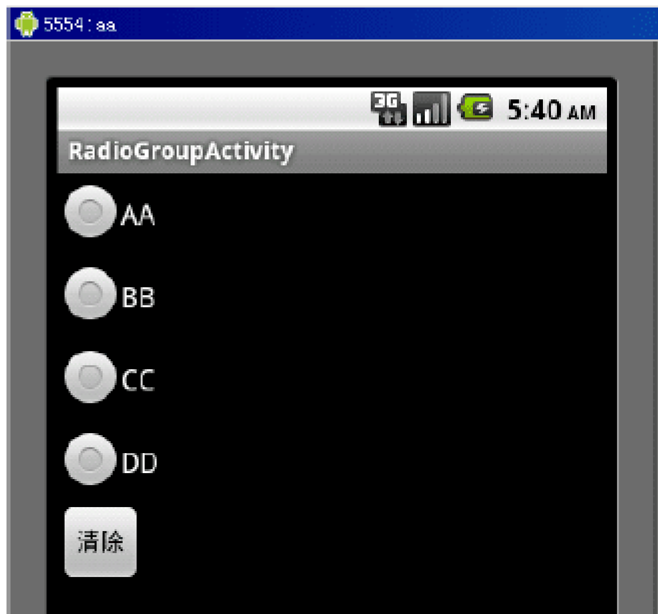


图 2-9　运行效果

2.8　使用 Spinner 控件实现下拉列表框效果

实例 016	在屏幕中显示下拉列表框
源码路径	光盘:\daima\009
视频路径	光盘:\视频\016
实例必备	016.Spinner 控件详解.pdf ① 概述 ② 重要属性 ③ 重要方法

2.8.1　实例说明

在 Android 手机屏幕中，可以使用下拉列表框控件 Spinner 来显示一个下拉列表框效果。使用下拉列表框后，用户不需要输入数据，只需选择一个选项后即可在框中完成数据输入工作。Spinner 位于

android.widget 包下，每次只显示用户选中的元素，当用户再次单击时，会弹出选择列表供用户选择，而选择列表中的元素同样来自适配器。其中，Spinner 是类 View 的一个子类。

2.8.2 具体实现

（1）在主布局文件 main.xml 中添加 Spinner 按钮，单击此按钮后会启动新界面 SpinnerActivity。

（2）在文件 MainActivity.java 中编写按钮单击事件的处理代码，具体如下所示。

```
private Button.OnClickListener spinner_button_listener = new Button.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setClass(MainActivity.this, SpinnerActivity.class);
                startActivity(intent);
            }
        };
```

在上述代码中，启动了 SpinnerActivity，此 SpinnerActivity 可以展示 Spinner 组件的界面。在具体实现上，首先创建了 SpinnerActivity 的 Activity，然后修改其 onCreate()方法，设置其对应模板为 spinner.xml。在文件 SpinnerActivity.java 中的对应代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setTitle("SpinnerActivity");
            setContentView(R.layout.spinner);
            find_and_modify_view();
        }
```

（3）编写文件下拉列表框界面的布局文件 spinner.xml，在其中添加 2 个 TextView 控件和 2 个 Spinner 控件。

（4）在设置文件 AndroidManifest.xml 中设置定义的 Spinner 组件的 ID 为 spinner_1，宽度占满了其父元素 LinearLayout 的宽，高度自适应。主要代码如下所示。

```
<activity android:name="SpinnerActivity"></activity>
```

经过上述操作后，就可以在屏幕中生成一个简单的单选选项界面，但是在列表中并没有选项值。如果要在下拉列表中实现可供用户选择的选项值，需要在里面填充一些数据。

（5）载入列表数据，首先定义需要载入的数据，然后在方法 onCreate()中通过调用 find_and_modify_view()来载入数据。在文件 SpinnerActivity.java 中实现上述功能的主要代码如下所示。

```
private static final String[ ] mCountries = { "China" ,"Russia", "Germany",
                "Ukraine", "Belarus", "USA" };
private void find_and_modify_view() {
            spinner_c = (Spinner) findViewById(R.id.spinner_1);
            allcountries = new ArrayList<String>();
            for (int i = 0; i < mCountries.length; i++) {
                allcountries.add(mCountries[i]);
            }
            aspnCountries = new ArrayAdapter<String>(this,
                    android.R.layout.simple_spinner_item, allcountries);
            aspnCountries
                    .setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            spinner_c.setAdapter(aspnCountries);
```

通过上述代码，将定义的 mCountries 数据载入到 Spinner 组件中。

（6）在文件 spinner.xml 中预定义数据，即在文件 spinner.xml 的模板中再添加一个 Spinner 组件。

（7）在文件 SpinnerActivity.java 中初始化 Spinner 中的值，具体代码如下所示。

```
spinner_2 = (Spinner) findViewById(R.id.spinner_2);
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
    this, R.array.countries, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner_2.setAdapter(adapter);
```

在上述代码中，将 R.array.countries 对应值载入到 spinner_2 中，在 R.array.countries 中对应的值是在文件 array.xml 中预先定义的。

执行后先显示两个下拉列表框，如图 2-10 所示；当单击一个下拉列表框后面的下拉三角按钮时会弹出一个由 Spinner 组件实现的下拉选项框，如图 2-11 所示；当选择一个选项后，选项值会自动出现在输入表单中，如图 2-12 所示。

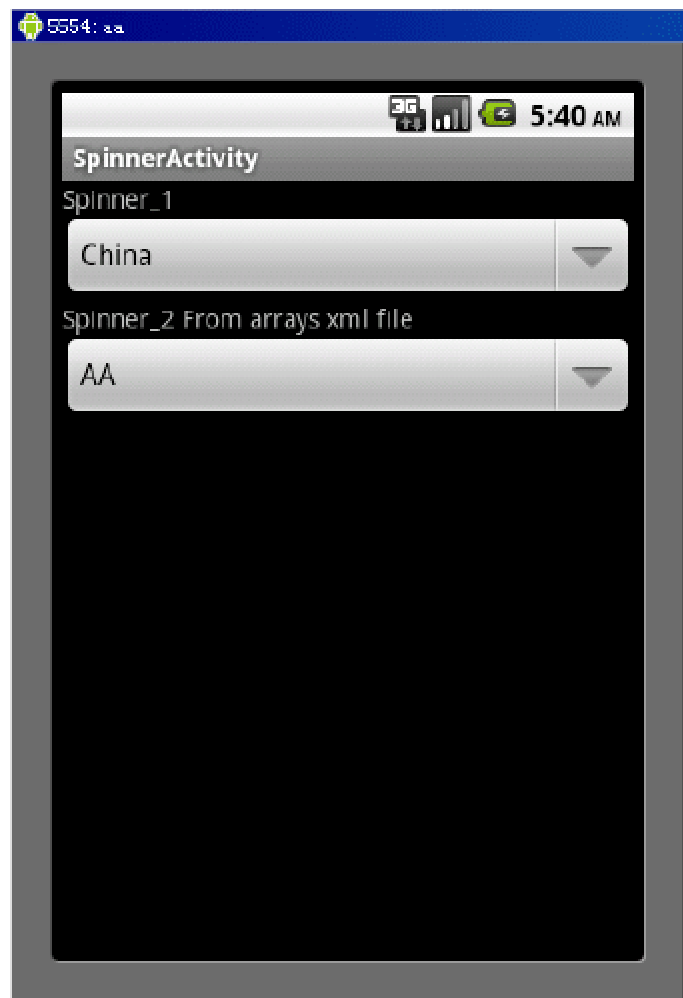


图 2-10　初始效果

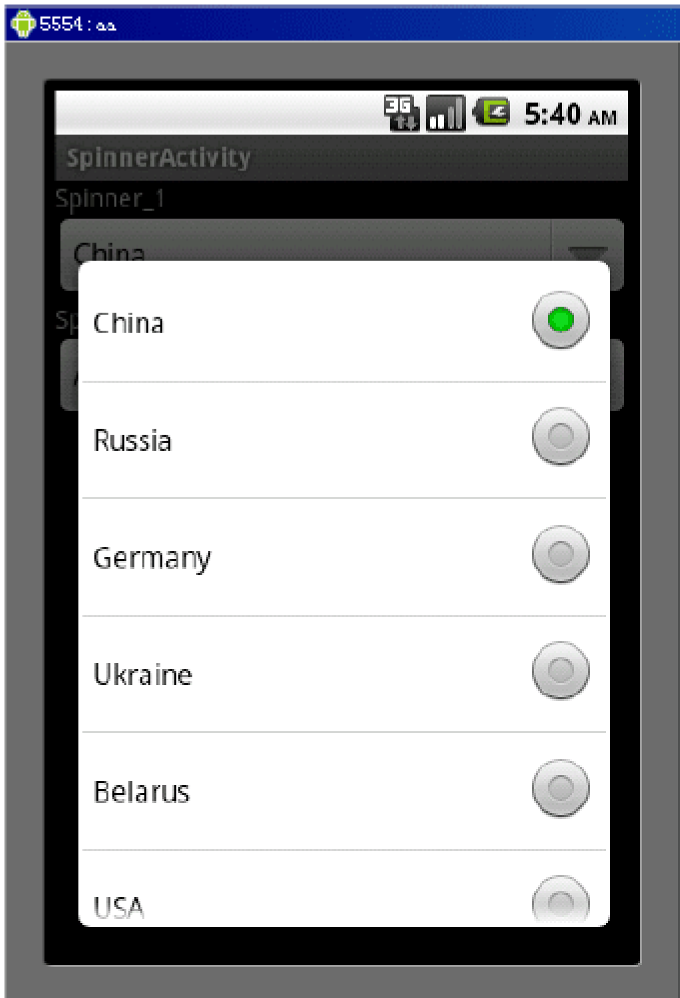


图 2-11　运行效果

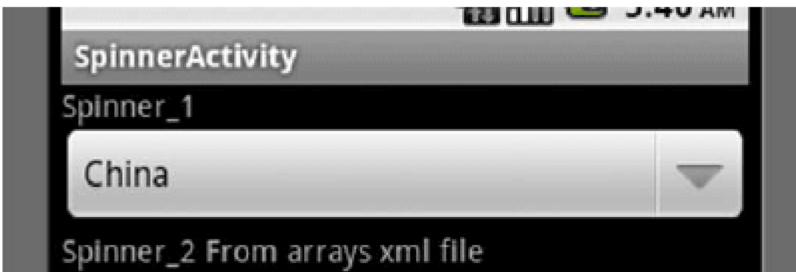


图 2-12　选择的值自动出现在表单中

2.9　使用 AutoCompleteTextView 控件自动输入文本

实例 017	在屏幕中实现自动输入文本效果
源码路径	光盘:\daima\009
视频路径	光盘:\视频\017
实例必备	017.AutoCompleteTextView 控件的属性.pdf

2.9.1　实例说明

在 Android 手机屏幕中，可以使用控件 AutoCompleteTextView 实现自动输入文本的功能。此控件

的主要功能是帮助用户自动输入数据，当用户输入一个字符后，能够根据这个字符提示显示出与之相关的数据。此应用在搜索引擎中比较常见，例如，用户在百度中输入关键字“客户”后，会在下拉列表中自动显示出相关的关键词，如图 2-13 所示。



图 2-13　百度的输入提示框

在控件 AutoCompleteTextView 中主要有如下 3 个方法。

☑ clearListSelection()：清除选中的列表项。

☑ dismissDropDown()：关闭下拉提示框菜单。

☑ getAdapter()：获取适配器。

2.9.2　具体实现

（1）修改主布局文件 main.xml，在其中添加 1 个 TextView 控件、1 个 AutoCompleteTextView 控件和 1 个 Button 控件。

（2）修改文件 MainActivity.java，添加自动完成功能处理事件的代码，主要代码如下所示。

```
/*定义要使用的类对象*/
private String[ ] normalString =
            new String[ ] {
              "Android", "Android Blog","Android Market", "Android SDK",
              "Android AVD","BlackBerry","BlackBerry JDE", "Symbian",
              "Symbian Carbide", "Java 2ME","Java FX", "Java 2EE",
              "Java 2SE", "Mobile", "Motorola", "Nokia", "Sun",
              "Nokia Symbian", "Nokia forum", "WindowsMobile", "Broncho",
              "Windows XP", "Google", "Google Android ", "Google  浏览器",
              "IBM", "MicroSoft", "Java", "C++", "C", "C#", "J#", "VB" };
@SuppressWarnings("unused")
private TextView show;
private AutoCompleteTextView autoTextView;
private Button clean;
private ArrayAdapter<String> arrayAdapter;
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      /*装入主屏布局 main.xml*/
```

```
            setContentView(R.layout.main);
            /*从 XML 中获取 UI 元素对象*/
            show = (TextView) findViewById(R.id.TextView_InputShow);
            autoTextView =
            (AutoCompleteTextView) findViewById(R.id.AutoCompleteTextView_input);
            clean = (Button) findViewById(R.id.Button_clean);
            /*实现一个适配器对象，用来给自动完成输入框添加自动装入的内容*/
            arrayAdapter = new ArrayAdapter<String>(this,
                        android.R.layout.simple_dropdown_item_1line, normalString);
            /*给自动完成输入框添加内容适配器*/
            autoTextView.setAdapter(arrayAdapter);
            /*给清空按钮添加点击事件处理监听器*/
            clean.setOnClickListener(new Button.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                        // TODO Auto-generated method stub
                        /*清空*/
                        autoTextView.setText("");
                  }
            });
      }
}
```

执行后可以在文本框中输入数据，输入后会根据预先准备的数据进行提示，如图 2-14 所示。

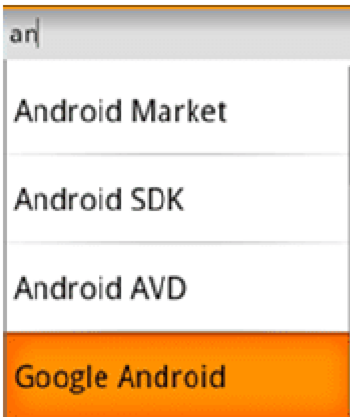


图 2-14　弹出文本输入提示框

2.10　使用日期选择器控件 DatePicker

实例 018	使用日期选择器控件 DatePicker
源码路径	光盘:\daima\009
视频路径	光盘:\视频\018
实例必备	018.控件 DatePicker 中的主要方法.pdf

2.10.1　实例说明

日期选择器控件 DatePicker 能够为用户提供快速选择日期的方法。日期的格式是“年-月-日”，在很多计算机系统和嵌入式系统中都为用户提供了日期选择表单，这样无需手工输入一个日期，只需利用鼠标单击就可以完成输入日期的工作。

2.10.2 具体实现

（1）在主布局文件 main.xml 中添加一个 ID 为 DatePicker_button 的按钮，单击后可以打开 DatePicker 日期界面。

（2）定义上述按钮响应处理事件，具体代码如下所示。

```
private Button.OnClickListener date_picker_button_listener = new Button.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setClass(MainActivity.this, DatePickerActivity.class);
            startActivity(intent);
        }
    };
```

当单击 DatePicker 按钮后会跳转到 DatePickerActivity 上。当创建一个 Activity 组件后，需要在其 onCreate()方法中指定需要绑定的模板文件为 date_picker.xml。

（3）在文件 DatePickerActivity.java 中编写 onCreate()方法的实现代码，设置默认的开始时间是 2011 年 5 月 17 日，主要代码如下所示。

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("CheckBoxActivity");
        setContentView(R.layout.date_picker);
        DatePicker dp =  (DatePicker)this.findViewById(R.id.date_picker);
        dp.init(2011, 5, 17, null);
    }
```

（4）在日期界面布局文件 date_picker.xml 中添加 DatePicker 组件，设置控件 DatePicker 的 ID 为 date_picker，其宽度和高度都为自适应。

（5）在文件 AndroidManifest.xml 中添加申明 Activity 的代码，具体代码如下所示。

```
<activity android:name="DatePickerActivity" />
```

执行后先显示设置的起始日期，如图 2-15 所示。分别单击月、日、年上面的“+”或下面的“-”后，将会自动显示更改后的月、日、年，如图 2-16 所示。

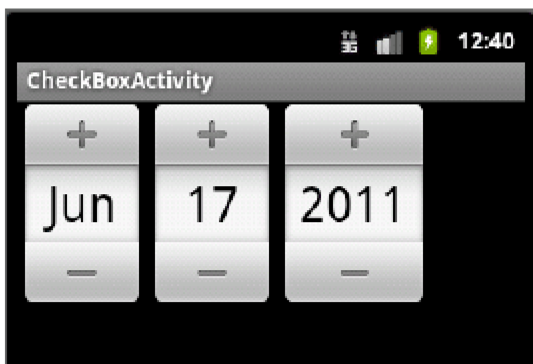


图 2-15 初始效果

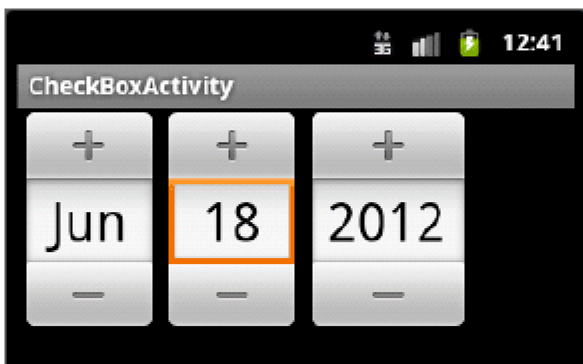


图 2-16 日期改变后的效果

2.11 使用时间选择器控件 TimePicker

实例 019	使用时间选择器控件 TimePicker
源码路径	光盘:\daima\009
视频路径	光盘:\视频\019
实例必备	019.DatePicker 和 TimePicker 的对比.pdf

2.11.1　实例说明

既然“年-月-日”格式的日期可以设置成自动输入，同理“时-分”格式的时间也可以设置成自动输入的方式。在 Android 系统中，可以使用控件 TimePicker 实现时间选择器的效果，其功能和控件 DatePicker 的功能类似，可以为用户提供快速选择时间的方法。

控件 DatePicker 继承自类 FrameLayout，如果要捕获用户修改日期选择控件中的数据事件，需要为 DatePicker 添加一个 OnDateChangedListener 监听器。

2.11.2　具体实现

（1）在文件 main.xml 中添加一个 Button 控件，单击后会来到“日期选择器”界面。

（2）编写单击按钮后响应事件的代码，当单击 time_picker 按钮后跳转到 TimePickerActivity。主要代码如下所示。

```
public void onClick(View v) {
        Intent intent = new Intent();
        intent.setClass(MainActivity.this, TimePickerTimePicker.class);
        startActivity(intent);
}
```

（3）创建一个新 Activity，在其 onCreate()方法中设置其对应的模板文件为 time_picker.xml，然后获取其中的 TimePicker 控件，主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setTitle("TimePickerActivity");
                setContentView(R.layout.time_picker);
                TimePicker tp =   (TimePicker)this.findViewById(R.id.time_picker);
                tp.setIs24HourView(true);
        }
```

（4）在文件 time_picker.xml 中定义控件 TimePicker。

执行后先显示设置的起始时间，单击时间上面的“+”或下面的“-”可以更改时间值，如图 2-17 所示。

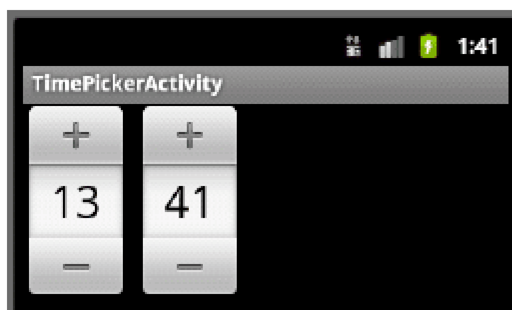


图 2-17　运行效果

2.12　使用 ScrollView 控件实现滚动效果

实例 020	在屏幕内实现滚动效果
源码路径	光盘:\daima\009
视频路径	光盘:\视频\020
实例必备	020.解决 Android 布局中 ScrollView 与 ListView 冲突的方法.pdf

2.12.1 实例说明

屏幕滚动功能十分重要，因为手机屏幕的大小有限，当需要显示很多信息时，可以通过滚动条来浏览所有的信息。在 Android 系统中，可以使用滚动视图控件 ScrollView 在手机屏幕中生成滚动样式的显示方式，这样即使内容超出了屏幕大小，也能通过滚动的方式供用户浏览。

2.12.2 具体实现

使用滚动视图控件 ScrollView 的方法比较简单，只需在 LinearLayout 外增加一个 ScrollView 标记即可，使用格式如下所示。

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
>
```

在上述代码中，将滚动视图控件 ScrollView 放在了 LinearLayout 的外面，这样当 LinearLayout 中的内容超过屏幕大小时，会实现滚动浏览功能。程序运行后的效果如图 2-18 所示。



图 2-18　运行效果

2.13 使用 ProgressBar 控件实现进度条效果

实例 021	在屏幕内实现进度条效果
源码路径	光盘:\daima\009
视频路径	光盘:\视频\021
实例必备	021.ProgressBar 控件详解.pdf ① XML 重要属性 ② 重要方法 ③ 重要事件

2.13.1 实例说明

在 Android 系统中，进度条控件 ProgressBar 可以用图像化的方式显示某个过程的进度，这样做的好处是能够更加直观地显示进度。进度条在计算机领域中非常常见，例如，软件安装过程一般都使用进度条。

2.13.2 具体实现

（1）在主布局文件 main.xml 中增加 ProgressBar 按钮。

（2）单击 ProgressBar 按钮后启动 ProgressBarActivity 以打开进度条界面，在文件 MainActivity.java 中对应的代码如下所示。

```
private Button.OnClickListener progress_bar_button_listener = new Button.OnClickListener() {
		public void onClick(View v) {
			Intent intent = new Intent();
			intent.setClass(MainActivity.this, ProgressBarActivity.class);
			startActivity(intent);
		}
	};
```

（3）编写文件 ProgressBarActivity.java 设置对应的布局文件为 Progress_Bar.xml，主要代码如下所示。

```
public class ProgressBarActivity extends Activity {
	CheckBox plain_cb;
	CheckBox serif_cb;
	CheckBox italic_cb;
	CheckBox bold_cb;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setTitle("ProgressBarActivity");
		setContentView(R.layout.progress_bar);
	}
```

（4）编写进度条界面的布局文件 Progress_Bar.xml，在其中插入如下两个 ProgressBar 控件。

☑ 第一个：环形进度条，进度到 50。

☑ 第二个：水平进度样式，进度到 75。

执行后将显示对应样式的进度条效果，如图 2-19 所示。

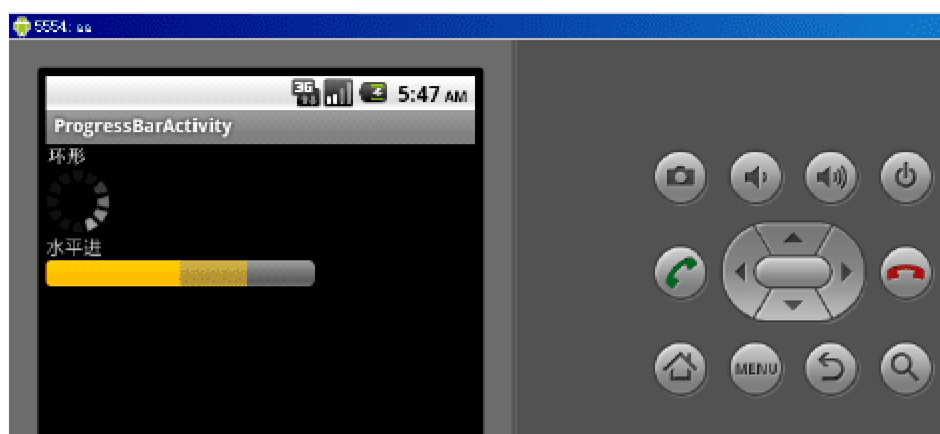


图 2-19 运行效果

2.14 使用 SeekBar 控件实现拖动条功能

实例 022	在屏幕内使用拖动条功能
源码路径	光盘:\daima\009
视频路径	光盘:\视频\022
实例必备	022.Android SeekBar 自定义 UI.pdf

2.14.1 实例说明

在 Android 系统中，使用拖动条控件 SeekBar 可以用拖动某个图标的方式来直观地显示某一进度。现实中最常见的拖动条应用是播放器的播放进度，用户可以通过拖动拖动条来控制播放进度。

2.14.2 具体实现

（1）在布局文件 main.xml 中插入 SeekBar 按钮。

（2）为 SeekBar 按钮编写处理事件代码，当用户单击按钮后会跳转到 SeekBarActivity。对应代码如下所示。

```
private Button.OnClickListener seek_bar_button_listener = new Button.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setClass(MainActivity.this, SeekBarActivity.class);
                startActivity(intent);
            }
        };
```

（3）为新建的 Activity 指定模板文件为 seek_bar.xml，在此文件中定义一个 SeekBar 控件，设置其 ID 为 Seek，设置宽度为布满屏幕显示，其最大值是 100。

（4）在文件 AndroidManifest.xml 中增加对 SeekBarActivity 权限的声明，对应代码如下所示。

```
<activity android:name="SeekBarActivity" />
```

执行后将显示对应样式的进度条，用户可以通过鼠标来拖动进度条的位置。如图 2-20 所示。

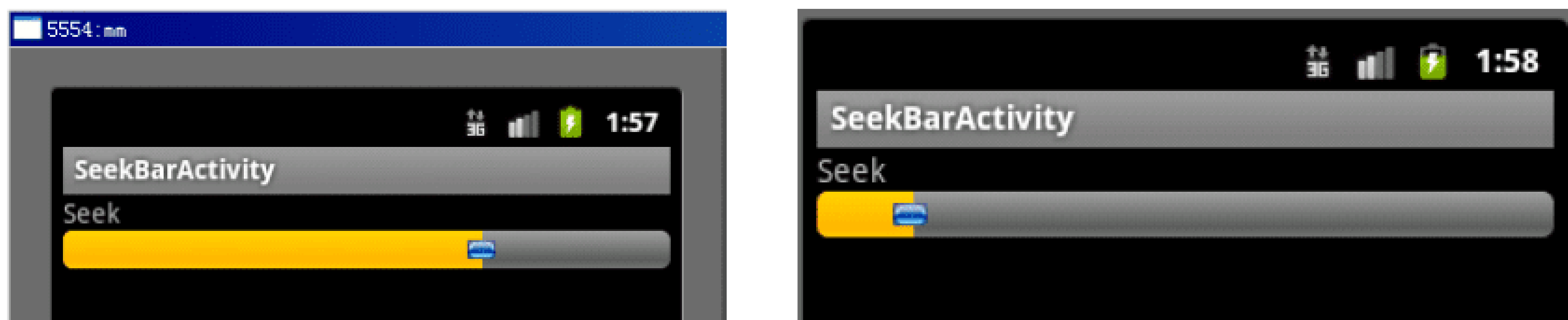


图 2-20　运行效果

2.15　使用评分组件 RatingBar

实例 023	在屏幕内使用评分组件
源码路径	光盘:\daima\009
视频路径	光盘:\视频\023
实例必备	023.Android 评分条 RatingBar 自定义设置.pdf

2.15.1 实例说明

在 Android 系统中，评分组件 RatingBar 的功能是为用户提供一个评分操作的模式。在日常应用中，

经常见到评分系统，例如，读者在卓越网或当当网购买图书后，可以对所购书发布评论并评分。使用 RatingBar 控件的流程非常清晰，其中最通用的流程如下所示。

（1）在布局文件中定义控件以及属性，这里主要需要指定的是总星星数量和当前的值，也就是总级别与当前级别的量，例如下面的代码。

```
<RatingBar
    android:id="@+id/ratingBar"
    android:numStars="5"                //总级别，总分，星星个数
    android:rating="1.5"                //当前级别，分数，星星个数
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</RatingBar>
```

（2）使用 RatingBar 控件中的方法实现评分，RatingBar 有如下两个比较重要的方法。

```
    RatingBar.setRating(float rating);  //设置评分
    RatingBar.getRating();              //获取评分
```

（3）使用如下事件监听处理完成评分操作。

```
RatingBar.setOnRatingBarChangeListener(new OnRatingBarChangeListener(){
    @Override
        public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {
                //doing actions
        }
    });
```

2.15.2　具体实现

（1）在文件 main.xml 中插入 RatingBar 按钮，具体代码如下所示。

```
<Button android:id="@+id/seek_bar_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RatingBar"
/>
```

（2）编写单击按钮的处理事件代码，当用户单击按钮后会跳转到 RatingBarActivity。对应代码如下所示。

```
private Button.OnClickListener rating_bar_button_listener = new Button.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setClass(MainActivity.this, RatingBarActivity.class);
            startActivity(intent);
        }
    };
```

（3）为新建的 Activity 指定模板文件为 rating_bar.xml，在文件中定义了一个 RatingBar 控件，设置其 ID 为 rating_bar，设置宽度和高度都是自适应。

（4）在文件 AndroidManifest.xml 中声明 RatingBarActivity 权限，对应代码如下所示。

```
<activity android:name="RatingBarActivity" />
```

执行后将显示对应样式的评级图，用户可以通过鼠标来选择评级，如图 2-21 所示。



图 2-21　运行效果

2.16　使用图片视图控件 ImageView

实例 024	在屏幕内显示一幅图片
源码路径	光盘:\daima\009
视频路径	光盘:\视频\024
实例必备	024.控件 ImageView 的常用属性.pdf

2.16.1　实例说明

在 Android 系统中，可以使用图片视图控件 ImageView 在屏幕中显示一幅图片。

2.16.2　具体实现

（1）在文件 main.xml 中插入 ImageView 按钮，具体代码如下所示。

```
<Button android:id="@+id/image_view_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="ImageView"
/>
```

（2）编写单击按钮的处理事件代码，当用户单击按钮后会跳转到 ImageViewActivity 界面。对应代码如下所示。

```
private Button.OnClickListener image_view_button_listener = new Button.OnClickListener() {
            public void onClick(View v) {
                  Intent intent = new Intent();
                  intent.setClass(MainActivity.this, ImageViewActivity.class);
                  startActivity(intent);
            }
      };
```

（3）为新建的 Activity 指定布局模板为 image_view.xml，在此设置 Android:src 为一张图片，该图片位于本项目根目录下的 res\drawable 文件夹中，支持 PNG、JPG、GIF 等常见的图片格式。

（4）编写文件 ImageViewActivity.java 文件，主要代码如下所示。

```
public class ImageViewActivity extends Activity {
      CheckBox plain_cb;
      CheckBox serif_cb;
      CheckBox italic_cb;
      CheckBox bold_cb;
```

```
        /** Called when the activity is first created.*/
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setTitle("ImageViewActivity");
                setContentView(R.layout.image_view);
//              find_and_modify_text_view();
        }
```

（5）在文件 AndroidManifest.xml 中声明权限 ImageViewActivity，对应代码如下所示。

```
<activity android:name="ImageViewActivity" />
```

执行后将在屏幕中显示指定的图片，如图 2-22 所示。



图 2-22 运行效果

2.17 使用图片按钮控件 ImageButton

实例 025	设置一幅图片当作按钮
源码路径	光盘:\daima\009
视频路径	光盘:\视频\025
实例必备	025.ImageButton 按下效果设计.pdf

2.17.1 实例说明

在 Android 系统中，为了使界面更加美观，通常使用在系统中将一幅图片作为按钮来使用，此功能是通过图片按钮控件 ImageButton 实现的。通过使用 ImageButton，可以使项目中的按钮更加美观大方。

2.17.2 具体实现

（1）在文件 main.xml 中插入 ImageButton 按钮，具体代码如下所示。

```
<Button android:id="@+id/image_button_button"
            android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:text="ImageButton" />
```

（2）编写单击按钮处理事件的代码，当用户单击按钮后会跳转到 ImageButtonActivity。对应代码如下所示。

```
private Button.OnClickListener image_button_button_listener = new Button.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setClass(MainActivity.this, ImageButtonActivity.class);
            startActivity(intent);
        }
    };
```

（3）为新建的 Activity 指定模板文件为 image_button.xml，在其中设置了 Android:src 的地址为一幅图片，该图片位于本项目根目录下的 res\drawable 文件夹中，支持 PNG、JPG、GIF 等常见的图片格式。

（4）编写对应的 Java 程序，对应代码如下所示。

```
public class ImageButtonActivity extends Activity {
    CheckBox plain_cb;
    CheckBox serif_cb;
    CheckBox italic_cb;
    CheckBox bold_cb;
    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("ImageButtonActivity");
        setContentView(R.layout.image_button);
//      find_and_modify_text_view();
    }
```

（5）在文件 AndroidManifest.xml 中声明 ImageButtonActivity 权限，对应代码如下所示。

```
<activity android:name="ImageButtonActivity" />
```

执行后将显示一个按钮，此按钮是使用指定的图片实现的，如图 2-23 所示。

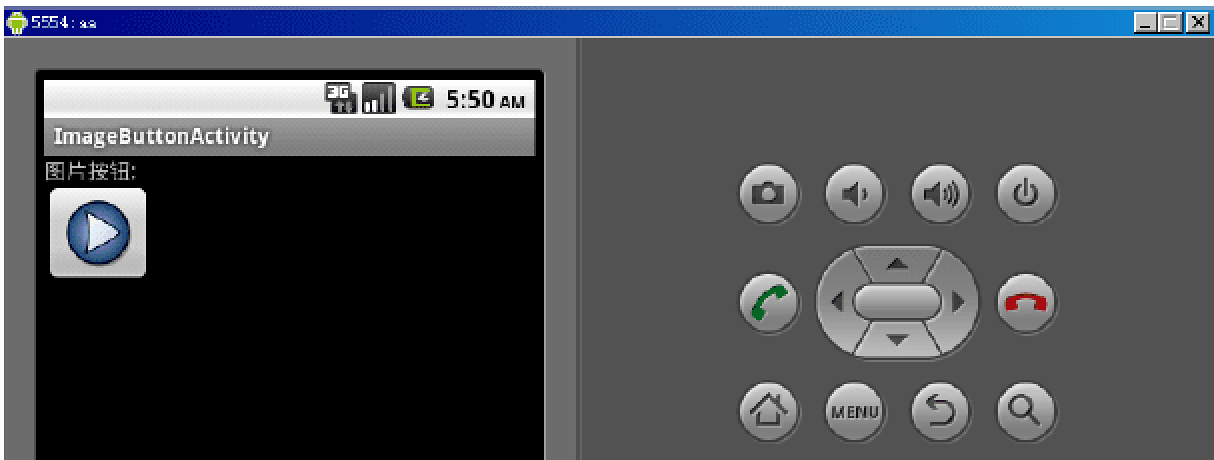


图 2-23　运行效果

2.18　使用 Gallery 控件实现类似 QQ 空间的照片效果

实例 026	实现类似 QQ 空间的照片效果
源码路径	光盘:\daima\009
视频路径	光盘:\视频\026
实例必备	026.使用 Gallery 控件实现个人相簿功能.pdf

2.18.1　实例说明

在 Android 系统中，可以使用切换图片控件实现和 QQ 空间中类似的幻灯图片显示效果，如图 2-24 所示。Android 系统中的切换图片控件有两个，分别是 ImageSwitcher 和 Gallery，其功能是以滑动的方式展现图片。在具体效果上，将首先显示一幅大图，然后在大图下面显示一组可以滚动的小图。

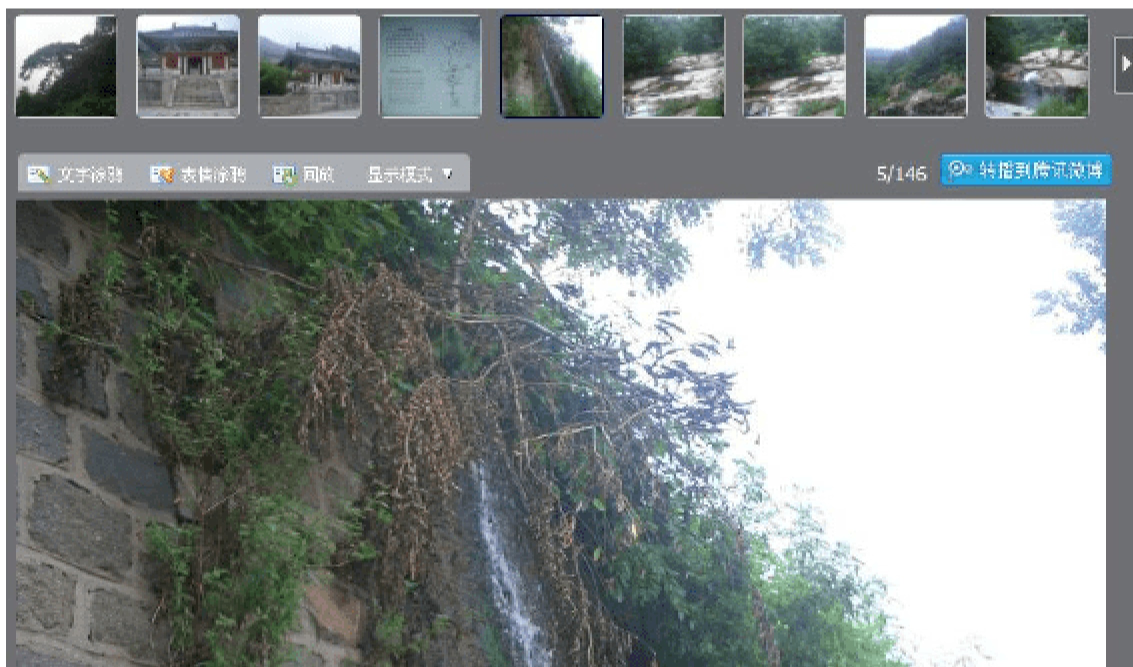


图 2-24　QQ 空间照片

2.18.2　具体实现

（1）编写单击按钮处理事件的代码，当用户单击按钮后会跳转到 ImageShowActivity。对应代码如下所示。

```
private Button.OnClickListener image_show_button_listener = new Button.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.setClass(MainActivity.this, ImageShowActivity.class);
                startActivity(intent);
            }
    };
```

（2）为新建的 Activity 指定模板文件为 image_show.xml，其中，在 RelativeLayout 中插入了两个控件，分别是 ImageSwitcher 和 Gallery。ImageSwitcher 用于显示上面的大图，Gallery 用于控制下面的小图列表索引。

（3）编写 Java 程序文件 ImageShowActivity.java，此文件的实现流程如下所示。

① 编写 onCreate()，使用 requestWindowFeature(Window.FEATURE_NO_TITLE)设置 Activity 没有 titlebar，只有这样，此图片的显示区域才会增大。而使用类 Gallery 的方法和使用 ListView 的方法相似，也需要使用 setAdapter 进行资源设置，对应的代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.image_show);
        setTitle("ImageShowActivity");

        mSwitcher = (ImageSwitcher) findViewById(R.id.switcher);
        mSwitcher.setFactory(this);
        mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
```

```
            android.R.anim.fade_in));
            mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));
            Gallery g = (Gallery) findViewById(R.id.gallery);
            g.setAdapter(new ImageAdapter(this));
            g.setOnItemSelectedListener(this);
    }
```

② 封装 BaseAdapter，通过 getView()方法来返回要显示的 ImageView，getView()方法的实现代码如下所示。

```
public View getView(int position, View convertView, ViewGroup parent) {
                ImageView i = new ImageView(mContext);
                i.setImageResource(mThumbIds[position]);
                i.setAdjustViewBounds(true);
                i.setLayoutParams(new Gallery.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
                i.setBackgroundResource(R.drawable.picture_frame);
                return i;
            }
```

通过上述代码动态生成了一个 ImageView，然后分别使用 setImageResource、setLayoutParams 和 setBackgroundResource 分别实现图片源文件、图片大小和图片背景的设置。当图片被显示到当前屏幕时，此函数会自动回调来提供要显示的 ImageView。

③ 在 ImageSwitcher1 中实现 ViewSwitcher.ViewFactory 接口，在此接口中定义方法 makeView()，为 ImageSwitcher 返回一个 View，在调用 ImageSwitcher 时，首先通过 Factory 为其提供一个 View，然后 ImageSwitcher 就可以初始化各种资源了，实现代码如下所示。

```
public View makeView() {
            ImageView i = new ImageView(this);
            i.setBackgroundColor(0xFF000000);
            i.setScaleType(ImageView.ScaleType.FIT_CENTER);
            i.setLayoutParams(new ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,
                        LayoutParams.FILL_PARENT));
            return i;
    }
```

（4）在文件 AndroidManifest.xml 中声明 Activity 权限，对应代码如下所示。

```
<activity android:name="ImageShowActivity" />
```

执行后将会按照 QQ 空间中的照片样式显示，如图 2-25 所示。

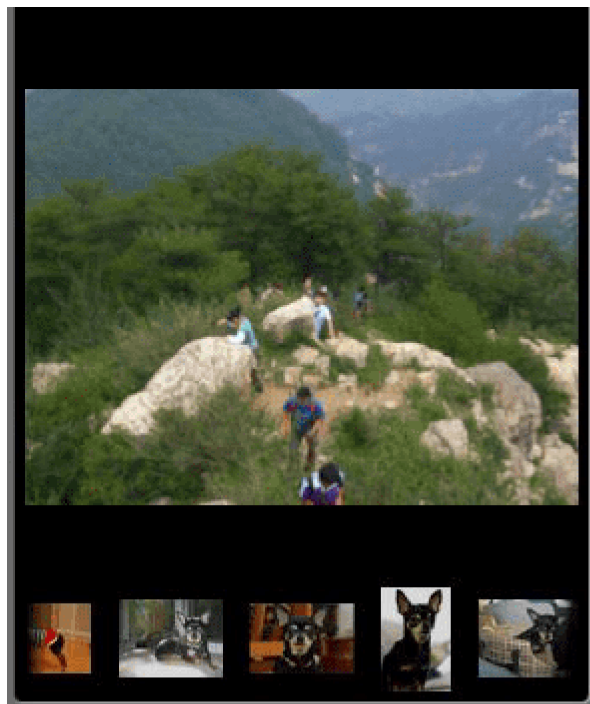


图 2-25　执行效果

2.19　使用网格视图控件 GridView

实例 027	使用网格视图控件布局屏幕
源码路径	光盘:\daima\009
视频路径	光盘:\视频\027
实例必备	027.升级实例.pdf

2.19.1　实例说明

随着手机系统的发展，大屏幕触摸手机受到了广大用户的追捧。其中最受欢迎的屏幕显示样式是以块状模式排列显示各个应用，如图 2-26 所示。



图 2-26　块状模式排列显示各个应用

在 Android 系统中，可以使用网格视图控件 GridView 将多幅指定的图片以指定大小、顺序排列的样式显示出来。此功能在相册的图片浏览中比较常见。

2.19.2　具体实现

（1）编写单击按钮的处理事件代码，当用户单击按钮后会跳转到 GridViewActivity，对应代码如下所示。

```
private Button.OnClickListener grid_view_button_listener = new Button.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setClass(MainActivity.this, GridViewActivity.class);
            startActivity(intent);
        }
    };
```

（2）在文件 GridViewActivity.java 中创建 onCreate()方法，为创建的 Activity 指定模板 grid_view.xml，然后获取其模板中的 GridView 控件，并使用 setAdapter()方法为其绑定一个合适的 ImageAdapter。具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.grid_view);
        setTitle("GridViewActivity");
        GridView gridview = (GridView) findViewById(R.id.grid_view);
        gridview.setAdapter(new ImageAdapter(this));
    }
```

（3）编写代码实现 ImageAdapter，因为 ImageAdapter 继承自 BaseAdapter，所以可以通过构造方法 ImageAdapter 获取 Context，然后实现了 getView，主要代码如下所示。

```
private Integer[ ] mThumbIds = {
        R.drawable.grid_view_01, R.drawable.grid_view_02,
```

```
            R.drawable.grid_view_03, R.drawable.grid_view_04,
            R.drawable.grid_view_05, R.drawable.grid_view_06,
            R.drawable.grid_view_07, R.drawable.grid_view_08,
            R.drawable.grid_view_09, R.drawable.grid_view_10,
            R.drawable.grid_view_11, R.drawable.grid_view_12,
            R.drawable.grid_view_13, R.drawable.grid_view_14,
            R.drawable.grid_view_15, R.drawable.sample_1,
            R.drawable.sample_2, R.drawable.sample_3,
            R.drawable.sample_4, R.drawable.sample_5,
            R.drawable.sample_6, R.drawable.sample_7
};
```

（4）在文件 AndroidManifest.xml 中声明 Activity，对应代码如下所示。

```
<activity android:name="GridViewActivity" />
```

执行后将会按照格子视图的方式显示指定的图片，如图 2-27 所示。

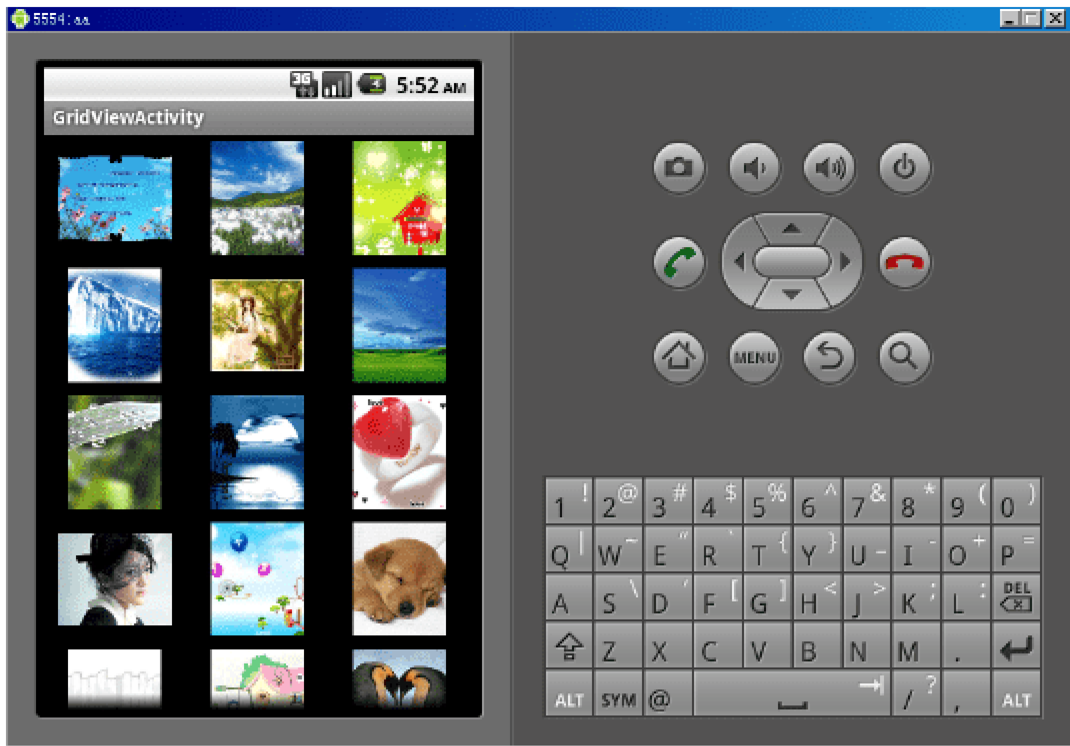


图 2-27　运行效果

2.20　使用 TabView 控件实现标签栏效果

实例 028	在屏幕内实现多个标签栏样式的效果
源码路径	光盘:\daima\009
视频路径	光盘:\视频\028
实例必备	028.TabView 的标准用法格式.pdf

2.20.1　实例说明

在 Android 系统中，可以使用标签控件 TabView 在屏幕内实现多个标签栏样式的效果，并且当单击某个标签栏时会打开对应的界面。

2.20.2　具体实现

（1）编写单击按钮处理事件代码，对应代码如下所示。

```
private Button.OnClickListener tab_demo_button_listener = new Button.OnClickListener() {
		public void onClick(View v) {
			Intent intent = new Intent();
			intent.setClass(MainActivity.this, TabDemoActivity.class);
			startActivity(intent);
		}
	};
```

（2）编写文件 TabDemoActivity.java 用于集成 TabActivity 并实现 TabDemoActivity，通过 getTabContentView 获取了 TabHost，然后绑定其模板，这样就绑定了每个标签的内容关联，主要代码如下所示。

```
@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setTitle("TabDemoActivity");
		TabHost tabHost = getTabHost();
		LayoutInflater.from(this).inflate(R.layout.tab_demo,
				tabHost.getTabContentView(), true);
		tabHost.addTab(tabHost.newTabSpec("tab1").setIndicator("tab1")
				.setContent(R.id.view1));
		tabHost.addTab(tabHost.newTabSpec("tab3").setIndicator("tab2")
				.setContent(R.id.view2));
		tabHost.addTab(tabHost.newTabSpec("tab3").setIndicator("tab3")
				.setContent(R.id.view3));
	}
```

（3）编写新界面模板文件 tab_demo.xml，在里面插入了 3 个 TextView 控件，当每个标签切换时，会显示各自对应的 TextView。

（4）在文件 AndroidManifest.xml 中声明 Activity 权限，对应代码如下所示。

```
<activity android:name="TabDemoActivity" />
```

执行后将会按指定的样式显示对应标签，如图 2-28 所示。

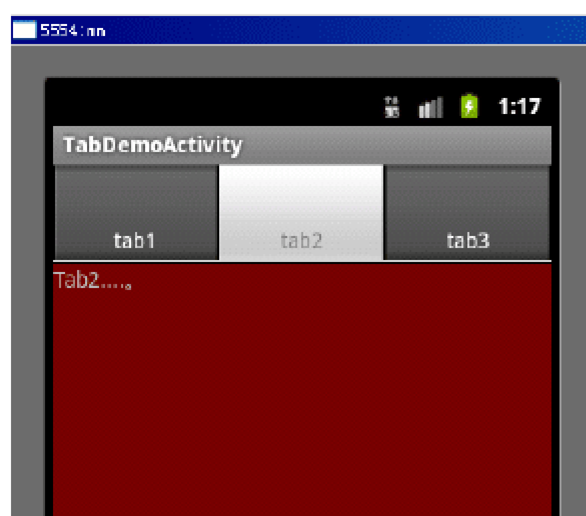


图 2-28　运行效果

2.21　使用 Toast 实现提醒

实例 029	使用 Toast 实现提醒
源码路径	光盘:\daima\029
视频路径	光盘:\视频\029
实例必备	029.Toast 的优势.pdf

2.21.1 实例说明

Toast 是 Android 系统中的一个提示对象，以简短小信息的样式提示用户某一种信息。提示信息以一个浮动的样式在屏幕的最上层显示，显示提示之后，静待几秒后便会自动消失。Toast 提示在项目中最常见的应用是调整音量大小，当单击音量调整钮之后，会看见跳出的音量指示 Toast 对象，等待调整完之后便会消失。

通过 Toast 的特性，可以在不影响用户通话或聆听音乐的情况下，显示要给 User 的信息。这样可以在任何程序运行时，通过 Toast 的方式显示运行变量或手机环境的概况。

在本实例中使用 EditText 控件接收用户输入的文字，当单击 Button 控件时，将 EditText 中的文字以 Toast.makeText()的方法显示于 Toast 对象中，这段文字在显示一段时间后自动消失。

2.21.2 具体实现

（1）在文件 main.xml 中分别插入 2 个 TextView 控件和 1 个 ImageButton 控件。

（2）在主程序文件中构建控件 EditText 与 Button 对象，并在 Button 的 onClick()方法中使用 Toast 对象的 makeText()方法显示输入的文字。实例文件的主要代码如下所示。

```
/** Called when the activity is first created.*/
/*声明两个对象变量（按钮与编辑文字）*/
private Button mButton;
private EditText mEditText;

@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*通过 findViewById()获取对象*/
  mButton=(Button)findViewById(R.id.myButton);
  mEditText=(EditText)findViewById(R.id.myEditText);

  /*设置 OnClickListener 给 Button 对象聆听 onClick 事件*/
  mButton.setOnClickListener(new OnClickListener()
  {
    @Override
    public void onClick(View v)
    {
    // TODO Auto-generated method stub

    /*声明字符串变量并取得用户输入的 EditText 字符串*/
    Editable Str;
    Str=mEditText.getText();

    /*使用系统标准的 makeText()方式产生 Toast 信息*/
    Toast.makeText(
      example3.this,
```

```
            "你的愿望   "+Str.toString()+"已送达宝贝信箱",
            Toast.LENGTH_LONG).show();

         /*清空 EditText*/
         mEditText.setText("");
         }
      });
   }
}
```

执行后的显示效果如图 2-29 所示，可以在文本框中输入祝福语句，然后单击 hello 按钮后发现在屏幕中出现提示一段信息，该提示信息会在一定时间内消失。Toast 构造参数中的第二个参数为显示的时间常数，可设置为 LENGTH_LONG 或 LENGTH_SHORT，前者提示时间较长，后者较短，作为传递 makeText()方法的参数使用，如图 2-30 所示。

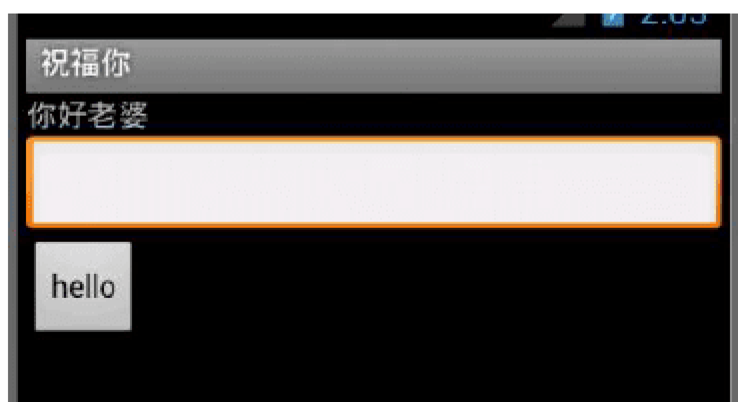


图 2-29　初始效果



图 2-30　温馨提示效果

2.22　在手机中实现文件搜索功能

实例 030	在手机中实现文件搜索功能
源码路径	光盘:\daima\030
视频路径	光盘:\视频\030
实例必备	030.搜索子目录.pdf

2.22.1　实例说明

在日常生活和学习过程中经常使用百度来搜索资料，而文件搜索是指通过输入关键字的方式快速检索需要的文件。同样，在 Android 手机系统中也可以使用文件搜索功能，此功能通过 Java I/O 的 API 中提供的 java.io.File 对象实现，只要利用 File 对象的方法，并结合 Android 中的对象 EditText 和 TextView 等就可以实现手机文件的搜索功能。

在本实例中，分别使用了 EditText、Button 和 TextView 这 3 个对象来实现此功能，用户将要搜索的文件名称或关键字输入 EditText 中，单击 Button 后，程序会在根目录中寻找符合的文件，并将搜索结果显示于 TextView 中；如果找不到符合的文件，则显示找不到文件。

2.22.2　具体实现

（1）在文件 main.xml 中分别插入 1 个 TextView 控件、1 个 EditText 控件和 1 个 Button 控件。

（2）编写主程序文件，以 java.io.File 对象来取得根目录下的文件，经过比较后，将符合结果的文件路径写入 TextView 中，若要在 TextView 中换行，需使用换行符“\n”实现换行处理。主程序文件的主要代码如下所示。

```
/*声明对象变量*/
private Button mButton;
private EditText mKeyword;
private TextView mResult;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  /*载入 main.xml Layout*/
  setContentView(R.layout.main);

  /*初始化对象*/
  mKeyword=(EditText)findViewById(R.id.mKeyword);
  mButton=(Button)findViewById(R.id.mButton);
  mResult=(TextView) findViewById(R.id.mResult);

  /*将 mButton 添加 onClickListener*/
  mButton.setOnClickListener(new Button.OnClickListener()
  {
    public void onClick(View v)
    {
      /*取得输入的关键字*/
      String keyword = mKeyword.getText().toString();
      if(keyword.equals(""))
      {
        mResult.setText("关键字不能为空!!");
      }
      else
      {
        mResult.setText(searchFile(keyword));
      }
    }
  });
}

/*搜索文件的 method*/
private String searchFile(String keyword)
{
  String result="";
  File[ ] files=new File("/").listFiles();
  for( File f : files )
  {
    if(f.getName().indexOf(keyword)>=0)
    {
      result+=f.getPath()+"\n";
```

```
        }
      }
      if(result.equals("")) result="找不到文件!!";
      return result;
    }
}
```

这样，整个实例介绍完毕。执行后的效果如图 2-31 所示，输入搜索关键字，并单击“检索文件”按钮后会在按钮下方显示对应的搜索结果，如图 2-32 所示。



图 2-31　执行效果



图 2-32　搜索结果

2.23　使用 AnalogClock 实现一个时钟效果

实例 031	使用 AnalogClock 实现一个时钟效果
源码路径	光盘:\daima\031
视频路径	光盘:\视频\031
实例必备	031.3 种常用的 System Clock.pdf

2.23.1　实例说明

如何编程实现一个时钟的界面效果呢？在 Android 系统中有一个专门的时钟对象 AnalogClock Widget，此对象可以实现时钟显示效果。在本实例屏幕的上方显示一个时钟效果界面，其下放置一个 TextView 以显示一个电子时钟效果。本实例需要使用如下 3 个对象。

- ☑ android.os.Handler：通过产生的 Thread 对象在进程内同步调用方法 System.currentTimeMillis()，这样可以取得系统时间。
- ☑ java.lang.Thread：是联系 Activity 与 Thread 的桥梁。
- ☑ android.os.Message：使用 Message 对象通知 Handler 对象，在收到 Message 对象后将时间变量的值显示在 TextView 中，这样就实现了数字时钟功能。

2.23.2　具体实现

编写主程序文件，在此需要另外加载 Java 对象 Calendar 和 Thread，在响应 onCreate()时构造这两个对象，并且实现了方法 handelMessage()和 run()。主程序文件的主要代码如下所示。

```
/*声明一常数作为判别信息*/
protected static final int GUINOTIFIER = 0x1234;
```

```
/*声明两个 Widget 对象变量*/
private TextView mTextView;
public AnalogClock mAnalogClock;

/*声明与时间相关的变量*/
public Calendar mCalendar;
public int mMinutes;
public int mHour;

/*声明关键 Handler 与 Thread 变量*/
public Handler mHandler;
private Thread mClockThread;

/** Called when the activity is first created.*/
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   /*通过 findViewById 取得两个 Widget 对象*/
   mTextView=(TextView)findViewById(R.id.myTextView);
   mAnalogClock=(AnalogClock)findViewById(R.id.myAnalogClock);

   /*通过 Handler 接收运行线程所传递的信息并更新 TextView*/
   mHandler = new Handler()
   {
      public void handleMessage(Message msg)
      {
         /*这里是处理信息的方法*/
         switch (msg.what)
         {
            case example13.GUINOTIFIER:
               /*在这处理要 TextView 对象 Show 时间的事件*/
               mTextView.setText(mHour+" : "+mMinutes);
               break;
         }
         super.handleMessage(msg);
      }
   };

   /*通过运行线程来持续取得系统时间*/
    mClockThread=new LooperThread();
    mClockThread.start();
}

/*改写一个 Thread Class 用来持续取得系统时间*/
class LooperThread extends Thread
{
   public void run()
   {
      super.run();
```

```
        try
        {
          do
          {
            /*取得系统时间*/
            long time = System.currentTimeMillis();
            /*通过 Calendar 对象取得小时与分钟*/
            final Calendar mCalendar = Calendar.getInstance();
            mCalendar.setTimeInMillis(time);
            mHour = mCalendar.get(Calendar.HOUR);
            mMinutes = mCalendar.get(Calendar.MINUTE);

            /*让运行线程休息 1 秒*/
            Thread.sleep(1000);
             /*重要关键程序：取得时间后发出信息给 Handler*/
             Message m = new Message();
             m.what = example13.GUINOTIFIER;
             example13.this.mHandler.sendMessage(m);
          }while(example13.LooperThread.interrupted()==false);
          /*当系统发出中断信息时停止本循环*/
        }
        catch(Exception e)
        {
          e.printStackTrace();
        }
      }
    }
}
```

执行后会显示一个数字时钟效果，如图 2-33 所示。

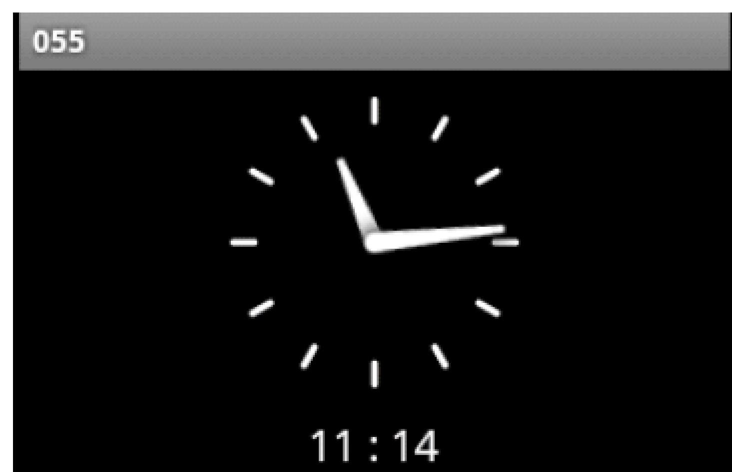


图 2-33　执行效果

2.24　实现不同的进度条效果

实例 032	在手机中实现不同的进度条效果
源码路径	光盘:\daima\032
视频路径	光盘:\视频\032
实例必备	032.在进度条中的 4 种不同风格.pdf

2.24.1 实例说明

在本实例中，使用 ProgressBar 和 Handler 联合实现了后台进度条提示效果。不但使用控件 ProgressBar 实现了进度条，而且使用 Handler 访问了新进程 Activity 中的 Widget，并将运行状态在屏幕中显示出来。通过 Handler 对象和 Message 对象，将进程中的状态向外传递，最后由 Activity 的 Handler 事件来取得运行状态。

2.24.2 具体实现

编写主程序文件，主要实现代码如下所示。

```
/*自定义 Handler 信息代码，用作识别事件处理*/
protected static final int GUI_STOP_NOTIFIER = 0x108;
protected static final int GUI_THREADING_NOTIFIER = 0x109;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mButton01 = (Button)findViewById(R.id.myButton1);
  mTextView01 = (TextView)findViewById(R.id.myTextView1);

  /*设置 ProgressBar widget 对象*/
  mProgressBar01 = (ProgressBar)findViewById(R.id.myProgressBar1);

  /*调用 setIndeterminate()方法赋值 indeterminate 模式为 false*/
  mProgressBar01.setIndeterminate(false);

  /*当单击按钮后，开始运行线程*/
  mButton01.setOnClickListener(new Button.OnClickListener()
  {
    @Override
    public void onClick(View v)
    {
      // TODO Auto-generated method stub

      /*单击按钮让 ProgressBar 显示出来*/
      mTextView01.setText(R.string.str_progress_start);

      /*将隐藏的 ProgressBar 显示出来*/
      mProgressBar01.setVisibility(View.VISIBLE);

      /*指定 Progress 最多为 100*/
      mProgressBar01.setMax(100);
```

```
        /*初始 Progress 为 0*/
        mProgressBar01.setProgress(0);

        /*开始一个运行线程*/
        new Thread(new Runnable()
        {
          public void run()
          {
            /*默认 0～9，共循环 10 次*/
            for (int i=0;i<10;i++)
            {
              try
              {
                /*成员变量，用以识别加载进度*/
                intCounter = (i+1)*20;
                /*每运行一次循环，即暂停 1 秒*/
                Thread.sleep(1000);

                /*当 Thread 运行 5 秒后显示运行结束*/
                if(i==4)
                {
                  /*以 Message 对象，传递参数给 Handler*/
                  Message m = new Message();

                  /*以 what 属性指定 User 自定义*/
                  m.what = example.GUI_STOP_NOTIFIER;
                  example.this.myMessageHandler.sendMessage(m);
                  break;
                }
                else
                {
                  Message m = new Message();
                  m.what = example.GUI_THREADING_NOTIFIER;
                  example60.this.myMessageHandler.sendMessage(m);
                }
              }
              catch(Exception e)
              {
                e.printStackTrace();
              }
            }
          }
        }).start();
      }
    });
  }

  /*Handler 构建之后，会聆听传来的信息代码*/
  Handler myMessageHandler = new Handler()
  {
    // @Override
```

```
public void handleMessage(Message msg)
{
  switch (msg.what)
  {
    /*当取得识别为离开运行线程时所取得的信息*/
    case example15.GUI_STOP_NOTIFIER:

      /*显示运行终了*/
      mTextView01.setText(R.string.str_progress_done);

      /*设置 ProgressBar Widget 为隐藏*/
      mProgressBar01.setVisibility(View.GONE);
      Thread.currentThread().interrupt();
      break;

      /*当取得识别为持续在运行线程当中时所取得的信息*/
      case example.GUI_THREADING_NOTIFIER:
      if(!Thread.currentThread().isInterrupted())
      {
        mProgressBar01.setProgress(intCounter);
        /*将显示进度显示于 TextView 中*/
        mTextView01.setText
        (
          getResources().getText(R.string.str_progress_start)+
          "("+Integer.toString(intCounter)+"%)\n"+
          "Progress:"+
          Integer.toString(mProgressBar01.getProgress())+
          "\n"+"Indeterminate:"+
          Boolean.toString(mProgressBar01.isIndeterminate())
        );
      }
      break;
  }
  super.handleMessage(msg);
}
```

通过上述代码，在屏幕中设计了一个按钮，单击此按钮后会显示 ProgressBar 进度条。在默认的布局文件 main.xml 中，并没有指定其 indeterm-inate 属性，所以即使在程序中调用了 ProgressBar 的 setindeterminate()方法，还是无法改变 ProgressBar.getProgress 的值，这个值永远都是 0。所以在上述代码中使用了循环动画作为运行过程中的显示素材，并使用了一个 counter 递增整数来表示运行进度的百分比。

执行后会显示一个按钮界面，用户单击“请按下”按钮后会显示一个提示进度条，如图 2-34 所示。

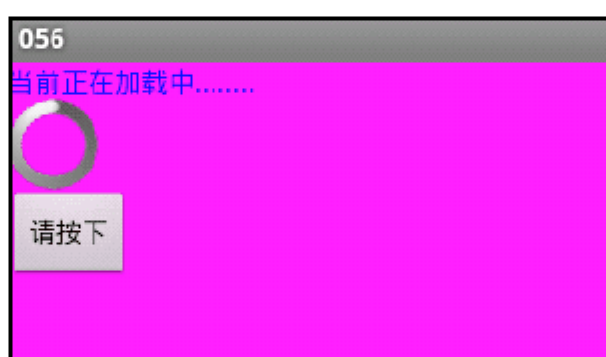


图 2-34　执行效果

2.25　使用 ListActivity 控件实现界面布局

实例 033	使用 ListActivity 实现界面布局
源码路径	光盘:\daima\033
视频路径	光盘:\视频\033
实例必备	033.ListActivity 的用法总结.pdf

2.25.1　实例说明

在 Andorid 系统中，ListActivity 是一个和 Activity 同级别的类。类 ListActivity 也能够实现布局处理，在显示菜单列表和列表明细项目方面比 Activity 更具有优势。

如果想让自己编写的程序继承于 ListActivity，可以通过下面的方法实现。

- ☑ getListAdapter()：获取列表项目的 Adapter。
- ☑ getListView()：获取列表的 View。
- ☑ getSelectItemId()：获取当前 Keypad 所选择的 Item ID。
- ☑ onContentChanged()：ListActivity 列表内容更新事件。
- ☑ onListItemClick(ListView,View,int,long)：User 在列表项目中单击触发事件。
- ☑ onRestoreInstancsState(Bundle)：恢复界面状态事件。
- ☑ setListAdapter(ListAdapter)：设置 ListAdapter 的列表项目。
- ☑ setSelection(int)：设置所选的项目。

2.25.2　具体实现

编写主程序文件实现信息显示功能，此文件的实现流程如下所示。

（1）声明对象变量，然后载入文件 main.xml 实现布局。

（2）设置项目的顺序位置。

（3）根据用户选择的选项显示出对应列表内的数据。

主程序文件的主要代码如下所示。

```
private int selectedItem = -1;
private String[ ] mString;
static final private int MENU_LIST1 = Menu.FIRST;
static final private int MENU_LIST2 = Menu.FIRST+1;
private ArrayAdapter<String> mla;

@Override
protected void onCreate(Bundle savedInstanceState)
{
  //调用父类的 onCreate 构造函数 savedInstanceState，保存当前 Activity 的状态信息
  super.onCreate(savedInstanceState);
```

```
}

@Override
protected void onListItemClick(ListView l, View v, int position, long id)
{
    //选择条目的位置
    selectedItem = position;
    Toast.makeText(shiyongListActivity.this, mString[selectedItem], Toast.LENGTH_SHORT).show();
    super.onListItemClick(l, v, position, id);
}

@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    // TODO Auto-generated method stub
    /*menu 组 ID*/
    int idGroup1 = 0;

    /*项目的顺序位置*/
    int orderMenuItem1 = Menu.NONE;
    int orderMenuItem2 = Menu.NONE+1;

    menu.add(idGroup1, MENU_LIST1, orderMenuItem1, R.string.str_menu_list1);
    menu.add(idGroup1, MENU_LIST2, orderMenuItem2, R.string.str_menu_list2);

    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    // TODO Auto-generated method stub
    switch(item.getItemId())
    {
        case (MENU_LIST1):
            mString = new String[ ]
                                 {
                                     getResources().getString(R.string.str_list1),
                                     getResources().getString(R.string.str_list2),
                                     getResources().getString(R.string.str_list3),
                                     getResources().getString(R.string.str_list4)
                                 };

            mla = new ArrayAdapter<String>(shiyongListActivity.this, R.layout.main, mString);
            shiyongListActivity.this.setListAdapter(mla);
            break;
        case (MENU_LIST2):
            mString = new String[ ]
                                 {
                                     getResources().getString(R.string.str_list5),
                                     getResources().getString(R.string.str_list6),
```

```
                                        getResources().getString(R.string.str_list7),
                                        getResources().getString(R.string.str_list8)
                                    };
            mla = new ArrayAdapter<String>(shiyongListActivity.this, R.layout.main, mString);
            shiyongListActivity.this.setListAdapter(mla);
            break;
        }
        return super.onOptionsItemSelected(item);
    }

}
```

执行后将在底部显示两个布局块，并分别显示设置的文本，具体效果如图 2-35 所示；单击布局块会弹出对应的信息，如图 2-36 和图 2-37 所示。

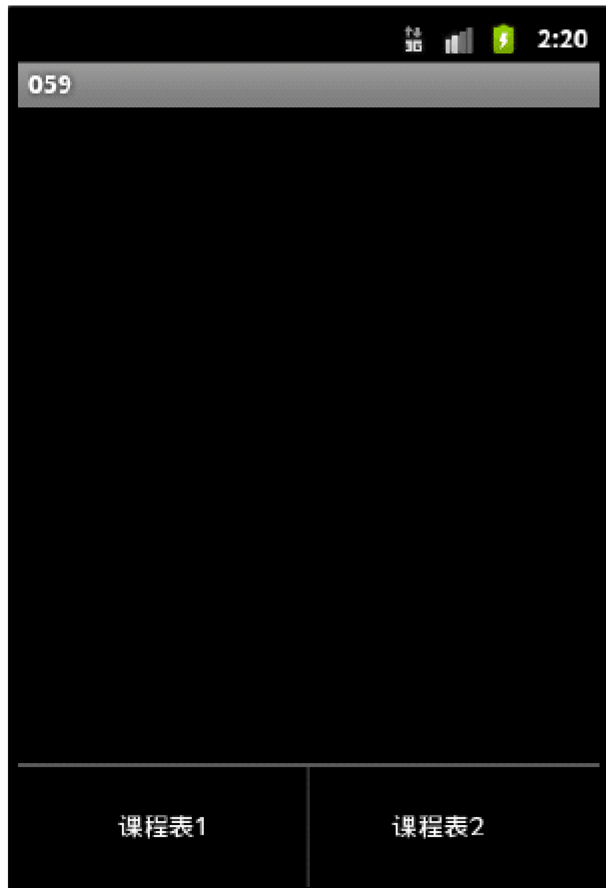


图 2-35　初始效果



图 2-36　第一个布局块对应的信息

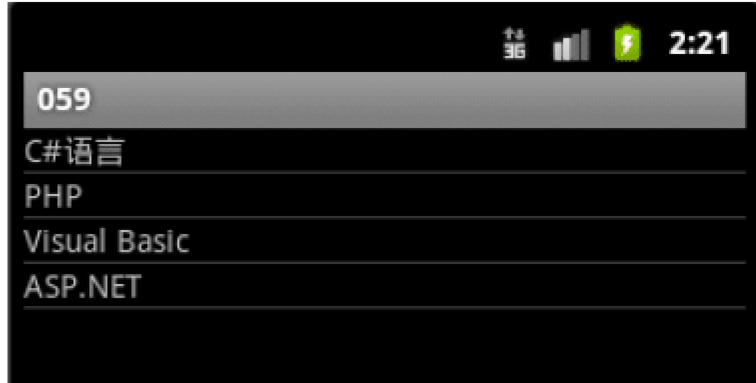


图 2-37　第二个布局块对应的信息

2.26　使用菜单控件 MENU

实例 034	使用菜单控件 MENU
源码路径	光盘:\daima\034
视频路径	光盘:\视频\034
实例必备	034.Android 系统中的 3 种菜单类型.pdf

2.26.1　实例说明

在 Android 系统中，可以使用控件 Menu 为用户提供一个友好的界面显示效果。大部分手机应用程序包括如下两种人机互动方式。

☑　第一种是直接通过 GUI 的 Views，可以满足大部分交互操作。

☑　第二种是应用 Menu，当单击 Menu 按钮后，会弹出与当前活动状态下的应用程序相匹配的菜单。

上述两种方式都有各自的优势，而且可以很好地相辅相成，即便用户可以由主界面完成大部分操作，但是适当拓展 Menu 功能可以使应用程序更加完善，至少用户可以通过排列整齐的按钮清晰地了解当前模式下可以使用的功能。

2.26.2 具体实现

1．编写布局文件 main.xml

在其中分别插入 1 个 TextView 控件和 2 个 Button 控件。其中 TextView 控件用于显示文本，并用 layout_width 设置了 Button 的宽度，用 layout_height 设置了 Button 的高度。通过符号@设置了读取变量值并进行替换。文件 main.xml 的主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/hello" />
    <Button android:id="@+id/button1"
        android:layout_width="100px"
        android:layout_height="wrap_content" android:text="@string/button1" />
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/button2" />
</LinearLayout>
```

☑ android:text="@string/button1"：相当于<string name="button1">button1</string>。

☑ android:text="@string/button2"：相当于<string name="button2">button2</string>。

上面的符号@十分重要，用于提示 XML 文件解析器解析@后面的名字，例如上面代码中的"@string/button1"，解析器会从文件 values\string.xml 中读取 Button1 这个变量值。

文件 string.xml 中定义了 TextView 和 Button 的值，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ActivityMenu</string>
    <string name="app_name">HelloMenu</string>
    <string name="button1">button1</string>
    <string name="button2">button2</string>
</resources>
```

2．编写主程序文件

首先定义函数 onCreate 来显示 main.xml 设置的布局，并设置两个 Button 为不可见状态；然后定义函数 onCreateOptionsMenu 来生成 Menu，此函数是一个回调方法，只有当按下手机设备上的 Menu 按钮后，Android 才会生成一个包含两个子项的菜单。在具体实现上，将首先得到 super 函数调用后的返回值，并在 onCreateOptionsMenu 的最后返回；然后调用 menu.add 给 menu 添加一个项；最后定义函数 onOptionsItemSelected，此函数是一个回调方法，只有当按下手机设备上的 Menu 按钮后，Android 才会调用执行。而这个事件就是单击菜单里的某一项，即 MenuItem。

主程序文件的主要代码如下所示。

```
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            button1 = (Button) findViewById(R.id.button1);
            button2 = (Button) findViewById(R.id.button2);
            /*设置两个 Button 不可见*/
            button1.setVisibility(View.INVISIBLE);
            button2.setVisibility(View.INVISIBLE);
                }
    @Override
    /*
     * menu.findItem(EXIT_ID);找到特定的 MenuItem
     * MenuItem.setIcon.可以设置 Menu 按钮的背景
     */
    public boolean onCreateOptionsMenu(Menu menu) {
            super.onCreateOptionsMenu(menu);
            menu.add(0, ITEM0, 0, "此时显示 button1");
            menu.add(0, ITEM1, 0, "此时显示 button2");
            menu.findItem(ITEM1);
            return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case ITEM0:
                    actionClickMenuItem1();
            break;
            case ITEM1:
                    actionClickMenuItem2(); break;

            }
            return super.onOptionsItemSelected(item);}
    /*
     *单击第一个 Menu 的第一个按钮执行的动作
     */
    private void actionClickMenuItem1(){
            setTitle("button1  可见");
            button1.setVisibility(View.VISIBLE);
            button2.setVisibility(View.INVISIBLE);
    }
    /*
     *  单击第二个 Menu 的第一个按钮执行的动作
     */
    private void actionClickMenuItem2(){
            setTitle("可以看到 button2");
            button1.setVisibility(View.INVISIBLE);
            button2.setVisibility(View.VISIBLE);
    }
}
```

执行后的效果如图 2-38 所示；当单击模拟器上的 Menu 键后会触发程序，并在屏幕中显示预先设置的已经隐藏的两个按钮，如图 2-39 所示。

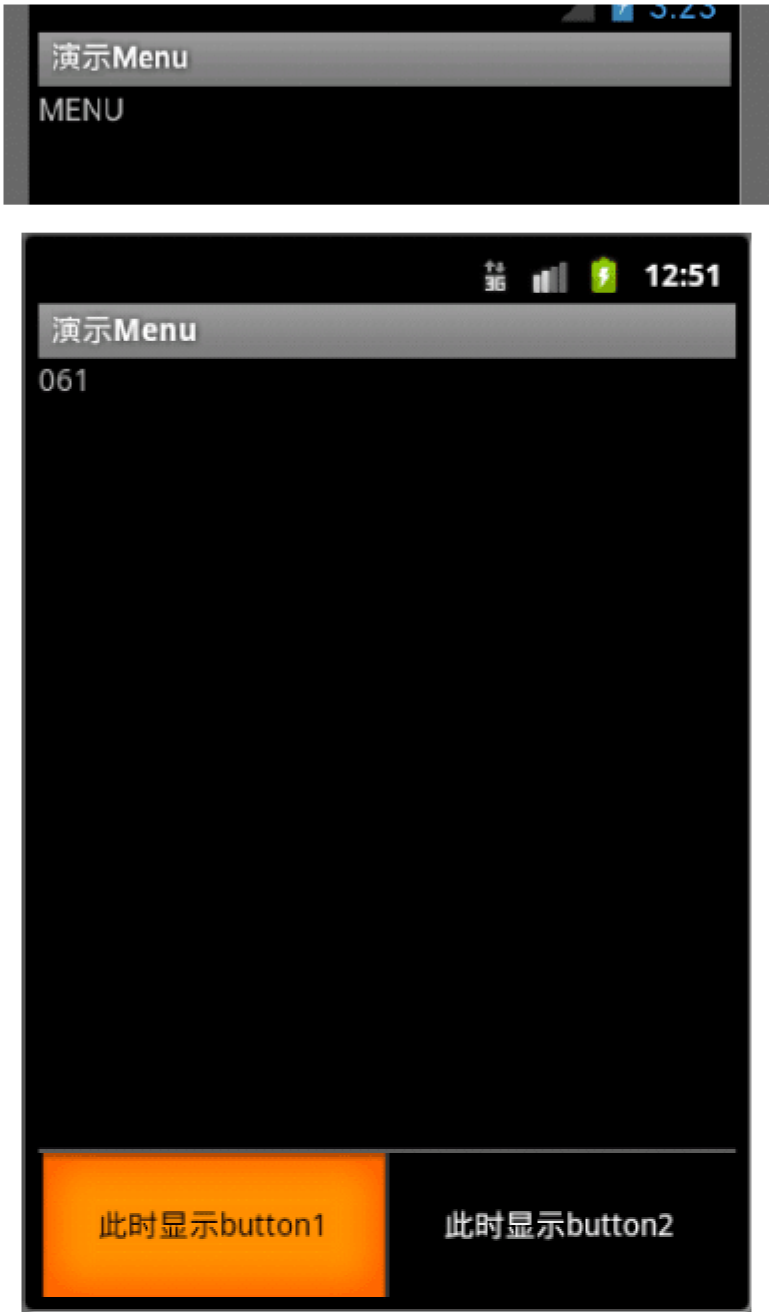


图 2-38　初始效果



图 2-39　触发设备后的效果

2.27　使用 SimpleAdapter 控件实现列表效果

实例 035	使用 SimpleAdapter 实现 ListView 效果
源码路径	光盘:\daima\035
视频路径	光盘:\视频\035
实例必备	035.ArrayAdapter 接受一个数组或者 List 作为参数.pdf

2.27.1　实例说明

在 Android 系统中，ListView 是最常用的组件之一，能够在屏幕内实现列表显示一些信息。ListView 通过 Adapter 适配器来构建显示功能，在 Android 系统中可以使用 3 种 Adapter，分别是 ArrayAdapter、SimpleAdapter 和 CursorAdapter。在本实例中，将使用 SimpleAdapter 来实现 ListView 效果。

2.27.2　具体实现

（1）构建 List 列表，设置用 Map 来实现列表中的每一项。然后编码创建 TestList（测试列表）类

继承 Activity，具体代码如下所示。

```
super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayList<HashMap<String, Object>> users = new ArrayList<HashMap<String, Object>>();
        for (int i = 0; i < 10; i++) {
            HashMap<String, Object> user = new HashMap<String, Object>();
            user.put("img", R.drawable.user);
            user.put("username", "姓名(" + i+")");
            user.put("age", (20 + i) + "");
            users.add(user);
        }
        SimpleAdapter saImageItems = new SimpleAdapter(this,
                users,              //数据来源
                R.layout.user,      //每一个 user xml 相当于 ListView 的一个组件
                new String[ ] { "img", "username", "age" },
                //分别对应 view 的 id
                new int[ ] { R.id.img, R.id.name, R.id.age });
        //获取 ListView
((ListView) findViewById(R.id.users)).setAdapter(saImageItems);
```

（2）编写文件 main.xml 实现布局，在其中插入 3 个 TextView，其中 ListView 前面是标题行，ListView 相当于用来显示数据的容器，里面每行是一个用户信息。

（3）编写文件 use.xml，用于定义用户信息布局。在文件中设置每行包含了一个 img 图片和两段 TextView 文字信息，这个文件以参数的形式通过 Adapter 在 ListView 中显示。

（4）编写主程序文件，使用 for 循环语句设置姓名是 i 递增，工资是 i*1000 递增，并且使用 SimpleAdapter 显示每块区域的用户信息，主要代码如下所示。

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayList<HashMap<String, Object>> users = new ArrayList<HashMap<String, Object>>();
        for (int i = 0; i < 10; i++) {
            HashMap<String, Object> user = new HashMap<String, Object>();
            user.put("img", R.drawable.user);
            user.put("username", "姓名(" + i+")");
            user.put("age", (1000 * i) + "");
            users.add(user);
        }
    SimpleAdapter saImageItems = new SimpleAdapter(this,
                users,                      //数据来源
                R.layout.user,              //每一个 user xml 相当于 ListView 的一个组件
                new String[ ] { "img", "username", "age" },
                //分别对应 view 的 id
                new int[ ] { R.id.img, R.id.name, R.id.age });
```

执行后的效果如图 2-40 所示。



图 2-40　运行效果

2.28　使用 Dialog 控件实现对话框效果

实例 036	在屏幕中演示使用多种对话框
源码路径	光盘:\daima\036
视频路径	光盘:\视频\036
实例必备	036.自定义消除 alertdialog 的黑、白边框.pdf

2.28.1　实例说明

对话框控件 Dialog 的功能是在手机屏幕中实现互动对话框效果。在本节的内容中，将通过一个具体实例的实现过程来讲解使用 Dialog 控件的方法。

2.28.2　具体实现

编写主程序文件，此文件比较复杂，下面详细讲解此文件的实现过程。

（1）定义方法 onCreateDialog()。

在此方法中使用 findViewById 通过组件的 ID 返回对该组件的引用，setOnClickListener 是单击 button1 按钮的监听器事件，onClick 为单击 Button 控件后的回调函数，函数 showDialog 是 Activity 中的函数，用于将 ID 为 DIALOG1 的 Dialog 显示出来。

onCreateDialog()方法的主要代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.alert_dialog);

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
```

```
                showDialog(DIALOG1);
            }
        });

        Button button2 = (Button) findViewById(R.id.buttons2);
        button2.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG2);
            }
        });

        Button button3 = (Button) findViewById(R.id.button3);
        button3.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG3);
            }
        });

        Button button4 = (Button) findViewById(R.id.button4);
        button4.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG4);
            }
        });
}
```

（2）编写方法 onCreateDialog()。

onCreateDialog()方法是一个回调函数，能够根据不同 Dialog 的 ID（编号）生成不同的 Dialog，例如，buildDialog1 函数用于生成第 1 个要显示的 Dialog，主要代码如下所示。

```
@Override
    protected Dialog onCreateDialog (int id) {
        switch (id) {
        case DIALOG1:
            return buildDialog1(ActivityMain.this);

        case DIALOG2:
            return buildDialog2(ActivityMain.this);

        case DIALOG3:
            return buildDialog3(ActivityMain.this);

        case DIALOG4:
            return buildDialog4(ActivityMain.this);

        }
        return null;
    }
```

（3）编写函数 buildDialog1、buildDialog2、buildDialog3 和 buildDialog4，这 4 个函数的实现原理是一样的，具体说明如下所示。

☑　buildDialog1：onClick()方法是监听器中的回调方法，当单击 Dialog 按钮时，系统会回调这个

方法。setNeutralButton()方法和 setPositiveButton()方法相对应，主要用于设置、取消按钮的一些属性。执行 builder.create 后，会生成一个配置好的 Dialog。

☑ buildDialog2：当单击第 2 个 Button 控件后会执行 buildDialog2，其中，方法 setNeutralButton()用于设置中间按钮的一些属性，具体设置方法与 buildDialog1 中一样。

☑ buildDialog3：当单击第 3 个 Button 控件后会执行 buildDialog3，其中通过 LayoutInflater 类的 inflater()方法，可以将一个 XML 布局变为一个 View 实例。下面的代码是整个 Dialog 的精髓。

```
builder.setView(textEntryView);
```

通过上述方法可以将实现好的个性化的 View 放置到 Dialog 中，此处的 textEntryView 与 alert_dialog_entry.xml 定义的布局相关联。

☑ buildDialog4：此程序最为简单，执行后将会显示一个等待界面。

上述 4 个函数的主要实现代码如下所示。

```
private Dialog buildDialog1(Context context) {
		/*创建一个 AlertDialog.Builder builder 对象*/
		AlertDialog.Builder builder = new AlertDialog.Builder(context);
		/*给 AlertDialog 预设一个图片*/
		builder.setIcon(R.drawable.alert_dialog_icon);
		/*给 AlertDialog 预设一个标题*/
		builder.setTitle(R.string.alert_dialog_two_buttons_title);
		/*设置按钮的属性*/
		builder.setPositiveButton(R.string.alert_dialog_ok,
				new DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog, int whichButton) {

						setTitle("您刚才单击了对话框上的“确定”按钮");
					}
				});
		builder.setNegativeButton(R.string.alert_dialog_cancel,
				new DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog, int whichButton) {

						setTitle("您刚才单击了对话框上的“取消”按钮");
					}
				});
		return builder.create();

	}
	private Dialog buildDialog2(Context context) {
		AlertDialog.Builder builder = new AlertDialog.Builder(context);
		builder.setIcon(R.drawable.alert_dialog_icon);
		builder.setTitle(R.string.alert_dialog_two_buttons_msg);
		builder.setMessage(R.string.alert_dialog_two_buttons2_msg);
		builder.setPositiveButton(R.string.alert_dialog_ok,
				new DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog, int whichButton) {

						setTitle("您刚才单击了对话框上的“确定”按钮");
					}
				});
```

```
        builder.setNeutralButton(R.string.alert_dialog_something,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {

                        setTitle("您刚才单击了对话框上的“进入详细”按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {

                        setTitle("您刚才单击了对话框上的“取消”按钮");
                    }
                });
        return builder.create();
    }

    private Dialog buildDialog3(Context context) {
        LayoutInflater inflater = LayoutInflater.from(this);
        final View textEntryView = inflater.inflate(
                R.layout.text_entry, null);//引用布局样式文件
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setIcon(R.drawable.alert_dialog_icon);
        builder.setTitle(R.string.alert_dialog_text_entry);
        builder.setView(textEntryView);
        builder.setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("您刚才单击了对话框上的“确定”按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("您刚才单击了对话框上的“取消”按钮");
                    }
                });
        return builder.create();
    }


    private Dialog buildDialog4(Context context) {
        ProgressDialog dialog = new ProgressDialog(context);
        dialog.setTitle("正在处理中");
        dialog.setMessage("等一会吧……");
        return dialog;
    }
}
```

执行后的初始效果如图 2-41 所示，单击第 1 个 Button 后的效果如图 2-42 所示，单击第 2 个 Button 后的效果如图 2-43 所示，单击第 3 个 Button 后的效果如图 2-44 所示，单击第 4 个 Button 后的效果如

图 2-45 所示。



图 2-41　初始效果



图 2-42　单击第 1 个 Button 后的效果



图 2-43　单击第 2 个 Button 后的效果



图 2-44　单击第 3 个 Button 后的效果

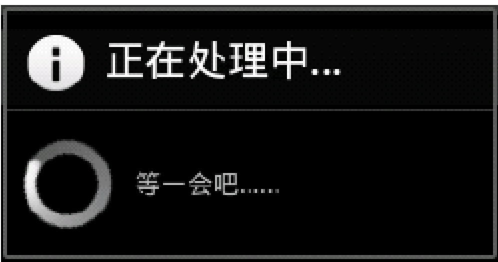


图 2-45　单击第 4 个 Button 后的效果

2.29　自定义一个 Android 控件

实例 037	自定义一个 Android 控件
源码路径	光盘:\daima\037
视频路径	光盘:\视频\037
实例必备	037.将属性值绑定到控件的基本步骤.pdf

2.29.1　实例说明

Android 应用项目离不开控件，Android 系统中提供了很多漂亮的控件，但控件虽多，功能却不强。例如，在 Radio 中只能放一个 text，而没有 value（key）可以存放，使得一个 RadioGroup 中的 RadioButton 可以同时被选择。如果是选择性别，将会出现错误。

Android 提供这些控件的目的是告诉用户如何实现一个最基本的控件，而非实现一个完整、强大的功能。Android 提供了自定义控件的功能，鼓励开发人员发挥聪明才智去开发自己的控件，可以在 Android 的基础控件上实现想要的功能。在本实例中，实现了自定义的组合控件——RadioButton 组合 RadioGroup。

2.29.2　具体实现

（1）设置自定义控件。在 Android 系统中自带的 RadioButton 只能存放 text，因为本实例的目标是

设计一个可以同时存放 key-value 对应的键值的 RadioButton，所以需要编写一个自定义控件来存放 key-value。实现思路非常简单，可以新建一个名为 ocom.sinxiao.view.MyRadioButton 的类，该类继承自 android.wedget.RadioButton，然后重写父类的所有构造方法。这样就实现了一个与父类一模一样的控件，然后在此基础上加入需要的功能即可实现一个功能更加强大的 RadioButton 控件，在本实例中加入一个属性 value，用来存放 RadioButton 的 key。

实现文件 MyRadioButton.java 的主要代码如下所示。

```
public class MyRadioButton extends android.widget.RadioButton implements OnCheckedChangeListener {
private String mValue;
public MyRadioButton(Context context, AttributeSet attrs, int defStyle) {
   super(context, attrs, defStyle);
}
public String getValue() {
   return this.mValue;
}
public void setValue(String value) {
   this.mValue = value;
}
public MyRadioButton(Context context, AttributeSet attrs) {
   super(context, attrs);
   try {
    /**
     * 与 values/attrs.xml 中定义的属性绑定
     */
    TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.RadioButton); this.mValue = a.getString
(R.styleable. RadioButton_value); a.recycle();
   } catch (Exception e) {
    e.printStackTrace();
   }
   setOnCheckedChangeListener(this);
}
public MyRadioButton(Context context) {
   super(context);
}
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
MyRadioGroup group = (MyRadioGroup) getParent();
group.setTheValue(this.getValue());
Log.d("-------Main", "the new value is ===>"+this.getValue());
}}
```

在上述代码中，加入一个新属性 value，由于 Android 习惯以 m 开头来命名属性，所以自定义控件也要遵循这个规则。而对于方法 setter()和 getter()来说，不需要加前缀 m。这样，就可以使用这个自定义控件了，而且可以为其设置一个 value，加上父类的 text 属性，就可以在 RadioButton 中加入 key- value 的键值了。

（2）在 XML 中引用自定义控件，只需要在控件名前加入包名即可。

（3）定义文件 attrs.xml 中的属性。既然在自定义控件中加入了一个新的属性，就应该能够在 XML 文件中引用并赋初始值。最简单的方法是在文件 res\values\attrs.xml 中定义属性，在自定义控件中获取

这个属性，然后与自定义控件的属性绑定。如果工程中没有文件 attrs.xml，则需要新建一个，在其中只存放自定义控件中需要的属性，此文件起到了一个中介的作用，负责将 layout\xx.xml 文件中对该变量的引用和自定义控件中的属性绑定起来。

（4）控件属性与 XML 的定义绑定。在文件 MyRadioButton.java 中存在如下代码。

```
/**
  * 与 values\attrs.xml 中定义的属性绑定
  */
TypedArray a = context.obtainStyledAttributes(attrs,
                    R.styleable.RadioButton);
 this.mValue = a.getString(R.styleable.RadioButton_value);
 a.recycle();
```

此处的 TypedArray 是一个存放资源的 Array 数组，能够从上下文中获取属性资源 R.styleable.RadioButton。其中 attrs 对应 attrs.xml 文件，由构造函数传进来。如下代码的作用是获取文件 attrs.xml 中定义的属性，并将该属性的值传给本控件的 mValue。

```
a.getString(R.styleable.RadioButton_value);
```

在最后返回一个绑定结束的信号给资源“a.recycle();”，整个绑定结束。

（5）在 XML 中对控件赋初始值。

绑定结束后需要在赋初始值处赋值，对控件赋初始值的代码如下所示。

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
xmlns:fsms="http://schemas.android.com/apk/res/com.sinxiao.myview"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
```

在上述代码中，首行声明命名空间，命名空间为 fsms，路径是 http://schemas.android.com/apk/res/，后面接的是 R 的路径 rog.kandy.R。然后在自定义控件的 XML 描述中，可以使用“fsms:value="true"”格式实现初始化赋值自定义控件的操作。

（6）实现 RadioGroup、RadioButton 组合控件的方法。

经过前面步骤的操作，仅仅是实现自定义控件，下面讲解实现组合控件的方法。在组合控件中，最常用到的就是 RadioGroup 和 RadioButton。RadioButton 的实现已经在前文介绍过了。下面要介绍 RadioGroup 的自定义控件和功能扩展。实现文件 MyRadioGroup.java 的主要代码如下所示。

```
        private String tag ="===myRadioGroup";
        //设置子控件的值
        private void setChildValue(){
          int n = this.getChildCount();
          Log.d(tag, "the n is "+n);
          for(int i=0;i<n;i++){
            MyRadioButton radio = (MyRadioButton)this.getChildAt(i);
            if(radio.getValue().equals(this.mValue)){
              radio.setChecked(true);
            }else{
              radio.setChecked(false);
            }
          }
        }
        //获取子类的值
        private void getChildValue(){
```

```
        int n = this.getChildCount();
        for(int i=0;i<n;i++){
          MyRadioButton radio = (MyRadioButton)this.getChildAt(i);
          if(radio.isChecked()){
            this.mValue=radio.getValue();
          }
        }
      }

      public void setTheValue(String value) {
        this.mValue = value;
      }

      public String getTheValue(){
        getChildValue();
        return this.mValue;
        }

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
            setChildValue();
    }
}
```

在上述代码中，RadioGroup 实现了如下两个功能。

☑　获取子控件（RadioButton）所选择的值。

☑　设置子控件要选择的值。

实现上述功能的方法非常简单，首先通过循环或者 RadioGroup 的子控件检测哪个控件被选择，然后通过 getValue 获取值，并将此 value 赋值给 RadioGroup 的扩展属性 value。当 MyRadioButton 发生改变时，向 RadioGroup 赋值，然后 RadioGroup 根据 value 值刷新界面，这样就实现了单选的效果。

执行后的效果如图 2-46 所示。



图 2-46　执行效果

2.30　设置控件的外观样式

实例 038	设置控件的外观样式
源码路径	光盘:\daima\038
视频路径	光盘:\视频\038
实例必备	038.查看 Android 开源代码.pdf

2.30.1　实例说明

在开发 Android 程序的过程中，系统提供的控件外观有时距离所要求的会有一定差距。此时可以

通过一些方法来改善控件的外观样式。在本实例中将详细讲解如何改变单选按钮控件 RadioButton 的外观样式。

2.30.2 具体实现

（1）首先制作一个 9patch 图片作为背景图，准备一幅 PNG 格式的图片，设置白色为透明色，如图 2-47 所示。

（2）使用 SDK/tools/draw9patch 工具在可伸缩的范围周围加上黑色的线告知系统这些区域可以伸缩。在处理后的图片周围多了黑色线，如图 2-48 所示。（注意：本实例后面有此工具的具体说明）



图 2-47　PNG 图片

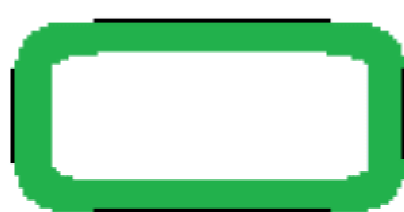


图 2-48　加黑线的图片

（3）继续使用 SDK/tools/draw9patch 工具制作按钮素材图片，具体说明如下所示。

☑ ：表示 enabled, on，紫色外框、红色中心点。

☑ ：表示 enabled, off，只有紫色外框。

☑ ：表示 enabled, on, pressed，黄色外框，红色中心点。

☑ ：表示 enabled, off, pressed，黄色外框。

☑ ：表示 disabled, on，灰色外框，灰色中心点。

☑ ：表示 disabled, off，灰色外框。

读者可以根据需要定义其他样式。

（4）使用 XML 文件描述一个 drawable，在 res\drawable\目录下创建文件 custom_radio_btn.xml。

（5）创建一个自定义的 Style 样式，并应用到 RadioButton 的 style 属性上，主要代码如下所示。

```
<style name="CustomRadioBtn">
    <item name="android:background">@drawable/radio_btn_bg</item>
    <item name="android:button">@drawable/custom_radio_btn</item>
</style>
```

此时设置 RadioButton 控件的外观功能就完成了，运行后的效果如图 2-49 所示。

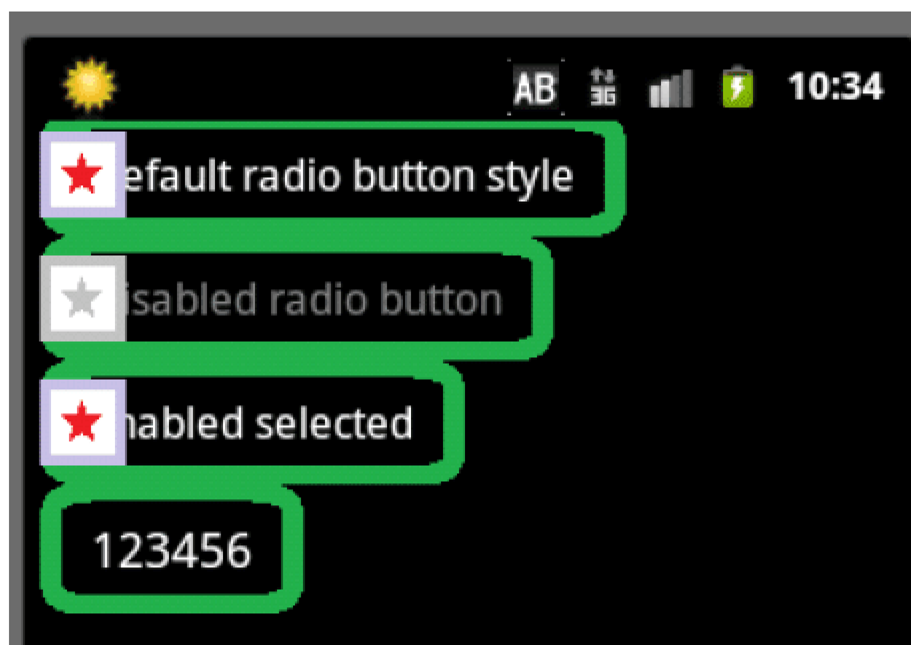


图 2-49　执行效果

2.31　使用 ExpandableListView 控件实现手风琴效果

实例 039	使用 ExpandableListView 实现手风琴效果
源码路径	光盘:\daima\039
视频路径	光盘:\视频\039
实例必备	039.ExpandableListAdapter 接口.pdf

2.31.1　实例说明

手风琴效果是网页中常见的一种特效，能够实现类似动感的手风琴效果，此功能经常用于新闻系统和信息列表展示项目中。在本实例中，将使用 ExpandableListView 在 Android 手机屏幕中实现一个手风琴效果。

在 ExpandableListView 中常用的方法如下所示。

☑ expandGroup(int groupPos)：在分组列表视图中展开一组。
☑ setSelectedGroup(int groupPosition)：设置选择指定的组。
☑ setSelectedChild(int groupPosition, int childPosition, boolean shouldExpandGroup)：设置选择指定的子项。
☑ getPackedPositionGroup(long packedPosition)：返回所选择的组。
☑ getPackedPositionForChild(int groupPosition, int childPosition)：返回所选择的子项。
☑ getPackedPositionType(long packedPosition)：返回所选择项的类型，如 Child 和 Group。
☑ isGroupExpanded(int groupPosition)：判断此组是否展开。

2.31.2　具体实现

主程序文件 ExpandableListDemo.java 的主要代码如下所示。

```
public class MyExpandableListAdapter extends BaseExpandableListAdapter {
        // Sample data set.   children[i] contains the children (String[ ]) for groups[i].
        public String[ ] groups = { "我的好友", "新疆同学", "亲戚", "同事" };
        public String[ ][ ] children = {
                { "胡算林", "张俊峰", "王志军", "二人" },
                { "李秀婷", "蔡乔", "别高", "余音" },
                { "摊派新", "张爱明" },
                { "马超", "司道光" }
        };

        public Object getChild(int groupPosition, int childPosition) {
            return children[groupPosition][childPosition];
        }
        public long getChildId(int groupPosition, int childPosition) {
```

```
            return childPosition;
        }
        public int getChildrenCount(int groupPosition) {
            return children[groupPosition].length;
        }
        public TextView getGenericView() {
            // Layout parameters for the ExpandableListView
            AbsListView.LayoutParams lp = new AbsListView.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT, 64);
            TextView textView = new TextView(ExpandableListDemo.this);
            textView.setLayoutParams(lp);
            // Center the text vertically
            textView.setGravity(Gravity.CENTER_VERTICAL | Gravity.LEFT);
            // Set the text starting position
            textView.setPadding(36, 0, 0, 0);
            return textView;
        }

        public View getChildView(int groupPosition, int childPosition, boolean isLastChild,
                View convertView, ViewGroup parent) {
            TextView textView = getGenericView();
            textView.setText(getChild(groupPosition, childPosition).toString());
            return textView;
        }
        public Object getGroup(int groupPosition) {
            return groups[groupPosition];
        }
        public int getGroupCount() {
            return groups.length;
        }
        public long getGroupId(int groupPosition) {
            return groupPosition;
        }
        public View getGroupView(int groupPosition, boolean isExpanded, View convertView,
                ViewGroup parent) {
            TextView textView = getGenericView();
            textView.setText(getGroup(groupPosition).toString());
            return textView;
        }
        public boolean isChildSelectable(int groupPosition, int childPosition) {
            return true;
        }
        public boolean hasStableIds() {
            return true;
        }
    }
```

执行后的效果如图 2-50 所示。



图 2-50　执行效果

2.32　使用 SlidingDrawer 控件实现滑动式抽屉效果

实例 040	使用控件 SlidingDrawer 在屏幕中实现滑动式抽屉的效果
源码路径	光盘:\daima\040
视频路径	光盘:\视频\040
实例必备	040.布局 SlidingDrawer 中的控件.pdf

2.32.1　实例说明

控件 SlidingDrawer 可以隐藏屏幕外的内容，并允许用户通过 handle 显示隐藏的内容。SlidingDrawer 可以垂直或水平滑动，由两个 View 组成，其中一个是可以拖动的 handle，另外一个是隐藏内容的 View。在本实例中，将使用控件 SlidingDrawer 在屏幕中实现滑动式抽屉的效果。

1．属性

☑　android:allowSingleTap：设置是否可以通过 handle 打开或关闭。
☑　android:animateOnClick：设置当使用者按下手柄打开/关闭时是否该有一个动画。
☑　android:content：隐藏的内容。
☑　android:handle：handle 手柄。

2．方法

☑　animateClose()：关闭时实现动画。
☑　close()：即时关闭。
☑　getContent()：获取内容。
☑　isMoving()：指示 SlidingDrawer 是否在移动。
☑　isOpened()：指示 SlidingDrawer 是否已全部打开。
☑　lock()：屏蔽触摸事件。
☑　setOnDrawerCloseListener(SlidingDrawer.OnDrawerCloseListener onDrawerCloseListener)：Sliding Drawer 关闭时调用。
☑　unlock()：解除屏蔽触摸事件。

☑ toggle()：切换打开和关闭的抽屉 SlidingDrawer。

2.32.2 具体实现

编写主程序文件 SlidingDrawerDemo.java，主要代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
   // TODO Auto-generated method stub
   super.onCreate(savedInstanceState);
   setContentView(R.layout.sildingdrawer);

   imbg=(ImageButton)findViewById(R.id.handle);
   mDrawer=(SlidingDrawer)findViewById(R.id.slidingdrawer);
   tv=(TextView)findViewById(R.id.tv);
   mDrawer.setOnDrawerOpenListener(new SlidingDrawer.OnDrawerOpenListener()
   {
     @Override
     public void onDrawerOpened() {
       flag=true;
       imbg.setImageResource(R.drawable.down);
     }
   });
   mDrawer.setOnDrawerCloseListener(new SlidingDrawer.OnDrawerCloseListener(){
     @Override
     public void onDrawerClosed() {
       flag=false;
       imbg.setImageResource(R.drawable.up);
     }
   });
   mDrawer.setOnDrawerScrollListener(new SlidingDrawer.OnDrawerScrollListener(){
     @Override
     public void onScrollEnded() {
       tv.setText("结束拖动");
     }
     @Override
     public void onScrollStarted() {
     tv.setText("开始拖动");
     }
   });
 }
```

执行后在屏幕中实现一个滑动式抽屉的效果，如图 2-51 所示。



图 2-51　执行效果

2.33 使用 ViewFlipper 控件实现左右滑动动画效果

实例 041	使用 ViewFlipper 控件实现左右滑动动画效果
源码路径	光盘:\daima\041
视频路径	光盘:\视频\041
实例必备	041.在 EditText 中插入 QQ 表情.pdf ① 实例说明 ② 具体实现 ③ 删除表情图片的思路

2.33.1 实例说明

在 Android 系统中，通过使用 ViewFlipper 控件可以在任意 View 之间实现切换。本实例的左右滑动功能主要是通过手势来控制的，手势向右滑动就调用 viewFlipper.showNext()方法，向左滑动就调用 viewFlipper.showPrevious()方法。

2.33.2 具体实现

（1）首先在布局文件 activity_main.xml 中添加 ViewFlipper 的标签，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
   <ViewFlipper
       android:id="@+id/viewFlipper"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       ></ViewFlipper>
</RelativeLayout>
```

（2）分别编写实现动画效果的如下 4 个文件。

☑ 文件 left_in.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="100%p"
        android:toXDelta="0"
        android:duration="600"
        />
    <alpha
        android:fromAlpha="0.1"
        android:toAlpha="1.0"
        android:duration="600"
        />

</set>
```

☑　文件 left_out.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="-100%p"
        android:duration="600"
        />
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.1"
        android:duration="600"
        />
</set>
```

☑　文件 right_in.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="-100%p"
        android:toXDelta="0"
        android:duration="600"
        />
    <alpha
        android:fromAlpha="0.1"
        android:toAlpha="1.0"
        android:duration="600"
        />
</set>
```

☑　文件 right_out.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="100%p"
        android:duration="600"
        />
    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.1"
        android:duration="600"
        />
</set>
```

（3）编写主程序文件 MainActivity.java，主要实现代码如下所示。

```
public class MainActivity extends Activity implements OnGestureListener {

    private static final String TAG = "MainActivity";

    private ViewFlipper viewFlipper;
    private GestureDetector detector;                    //手势检测
```

```
 Animation leftInAnimation;
 Animation leftOutAnimation;
 Animation rightInAnimation;
 Animation rightOutAnimation;
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      requestWindowFeature(Window.FEATURE_NO_TITLE);
      setContentView(R.layout.activity_main);

      viewFlipper = (ViewFlipper)findViewById(R.id.viewFlipper);
      detector = new GestureDetector(this);

      //向 viewFlipper 添加 View
      viewFlipper.addView(getImageView(R.drawable.new_feature_1));
      viewFlipper.addView(getImageView(R.drawable.new_feature_2));
      viewFlipper.addView(getImageView(R.drawable.new_feature_3));
      viewFlipper.addView(getImageView(R.drawable.new_feature_4));
      viewFlipper.addView(getImageView(R.drawable.new_feature_5));
      viewFlipper.addView(getImageView(R.drawable.new_feature_6));

      //动画效果
        leftInAnimation = AnimationUtils.loadAnimation(this, R.anim.left_in);
        leftOutAnimation = AnimationUtils.loadAnimation(this, R.anim.left_out);
        rightInAnimation = AnimationUtils.loadAnimation(this, R.anim.right_in);
        rightOutAnimation = AnimationUtils.loadAnimation(this, R.anim.right_out);
}

  private ImageView getImageView(int id){
  ImageView imageView = new ImageView(this);
  imageView.setImageResource(id);
  return imageView;
}

@Override
public boolean onTouchEvent(MotionEvent event) {

        return this.detector.onTouchEvent(event);                //touch 事件交给手势处理
}

 @Override
 public boolean onDown(MotionEvent e) {
        // TODO Auto-generated method stub
        return false;
 }

 @Override
 public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
               float velocityY) {
        Log.i(TAG, "e1="+e1.getX()+" e2="+e2.getX()+" e1-e2="+(e1.getX()-e2.getX()));
        if(e1.getX()-e2.getX()>120){
```

```
                viewFlipper.setInAnimation(leftInAnimation);
                viewFlipper.setOutAnimation(leftOutAnimation);
                viewFlipper.showNext();             //向右滑动
                return true;
            }else if(e1.getX()-e2.getY()<-120){
                viewFlipper.setInAnimation(rightInAnimation);
                viewFlipper.setOutAnimation(rightOutAnimation);
                viewFlipper.showPrevious();         //向左滑动
                return true;
            }
            return false;
        }
        @Override
        public void onLongPress(MotionEvent e) {
            // TODO Auto-generated method stub

        }
        @Override
        public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
                float distanceY) {
            // TODO Auto-generated method stub
            return false;
        }
        @Override
        public void onShowPress(MotionEvent e) {
            // TODO Auto-generated method stub

        }
        @Override
        public boolean onSingleTapUp(MotionEvent e) {
            // TODO Auto-generated method stub
            return false;
        }
}
```

执行后的效果如图 2-52 所示。



图 2-52　执行效果

第 3 章　事件处理实战

与界面编程最紧密相关的知识就是事件处理了，当用户在程序界面上执行各种操作时，应用程序必须为用户动作提供响应，这种响应动作需要通过事件处理来完成。Android 系统提供了两种事件处理的方式，分别是基于回调的事件处理和基于监听器的事件处理。本章将通过具体的实例来讲解开发 Android 系统事件处理应用程序的基本用法，为读者学习本书后面的知识打下基础。

3.1　使用 setOnKeyListener 事件实现文本处理

实例 042	使用 EditText 控件和 setOnKeyListener 事件实现文本处理
源码路径	光盘:\daima\042
视频路径	光盘:\视频\042
实例必备	042.基于监听的事件处理.pdf ① 监听处理模型中的 3 种对象 ② Android 系统中的监听事件 ③ 实现事件监听器的方法

3.1.1　实例说明

通过该实例将演示使用 EditText 和 setOnKeyListener 实现文本处理的方法。在实例中将以 EditText 和 TextView 来演示如何捕捉用户在键盘中输入的光标，同时获取输入的文字，并且同步显示于 TextView，类似手机版的 Ajax 效果，实现实时输入并实时输出的效果。

3.1.2　具体实现

（1）在文件 main.xml 中分别插入 2 个 TextView 控件和 1 个 EditText 控件。

（2）编写主程序文件，此文件的重点是用 EditText.OnKeyListener 拦截在 EditText 中的输入事件，只需要重写 onKey()方法即可实现这个功能。在 onKey()方法中，将从 EditText.getText()中获取的文字显示在 TextView 中，此文件的主要代码如下所示。

```
/*声明 TextView、EditText 对象*/
private TextView mTextView01;
private EditText mEditText01;

/** Called when the activity is first created.*/
  @Override
public void onCreate(Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /*取得 TextView、EditText*/
        mTextView01 = (TextView)findViewById(R.id.myTextView);
        mEditText01 = (EditText)findViewById(R.id.myEditText);

        /*设置 EditText 用 OnKeyListener 事件来启动*/
        mEditText01.setOnKeyListener(new EditText.OnKeyListener()
        {
            @Override
            public boolean onKey(View arg0, int arg1, KeyEvent arg2)
            {
                // TODO Auto-generated method stub
                /*设置 TextView 显示 EditText 所输入的内容*/
                mTextView01.setText(mEditText01.getText());
                return false;
            }
        });
    }
```

执行后的效果如图 3-1 所示，当在文本框中输入字符后，可以在文本框下方即时显示输入的字符，如图 3-2 所示。



图 3-1　初始效果

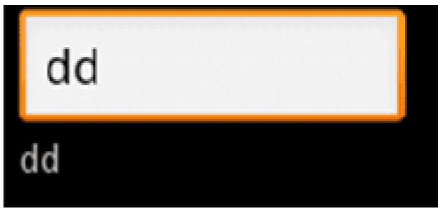


图 3-2　即时显示提示信息

3.2　实现一个有背景图片的按钮

实例 043	在屏幕中实现一个背景图片按钮
源码路径	光盘:\daima\043
视频路径	光盘:\视频\043
实例必备	043.基于回调的事件处理.pdf ① Android 事件侦听器的回调方法 ② 基于回调的事件传播 ③ 重写 onTouchEvent()方法响应触摸屏事件

3.2.1　实例说明

在现实应用中，为了特殊需要，需要设计一个具有背景图的按钮，让按钮有美观的背景图片。在 Android 的手机屏幕中，也可以实现一个具有背景图的按钮，具体设计流程如下所示。

（1）将按钮背景图预先导入至 Drawable 中（*.png 图形文件），利用这些图片作为 ImageButton 的背景图。

（2）在 Layout 中配置一个“一般按钮”，读者可以查看两者的对照效果，在运行之后可以明显地看出图片按钮与一般按钮在外观上存在的差异。

有许多方法可以设置 ImageButton 背景图，在本实例中使用了 ImageButton.setImageResource()方法来实现，在此方法中需要传递的参数是 res/drawable/Resource ID。除了设置上述方法外，还需要使用 onFocusChange 与 onClick 等按钮事件来处理单击按钮之后的操作，最后通过 TextView 显示目前图片按钮的状态为 onClick、onFocus 或 offFocus，并且同步更新按钮的背景图，让用户感觉有动态交互的感觉。

3.2.2　具体实现

（1）在布局文件 main.xml 中分别插入 1 个 TextView 控件、1 个 ImageButton 控件和 1 个 ImageButton 控件，分别构造 ImageButton、Button 和 TextView 这 3 个对象，并在 ImageButton 上设置 onFocusChangeListener 与 onClickListener 事件，这样就实现了 ImageButton 图片的置换功能。

（2）编写主程序文件 example2.java，在 ImageButton 上设置按钮事件 onFocusChangeListener 与 onClickListener 的处理方法，并实现 ImageButton 图片的置换，实例文件的主要代码如下所示。

```
/*声明 3 个对象变量（图片按钮、按钮与 TextView）*/
private ImageButton mImageButton1;
private Button mButton1;
private TextView mTextView1;

/*Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*通过 findViewById 建构 3 个对象*/
  mImageButton1 =(ImageButton) findViewById(R.id.myImageButton1);
  mButton1=(Button)findViewById(R.id.myButton1);
  mTextView1 = (TextView) findViewById(R.id.myTextView1);

  /*通过 OnFocusChangeListener 来应答 ImageButton 的 onFocus 事件*/
  mImageButton1.setOnFocusChangeListener(new OnFocusChangeListener()
  {
    public void onFocusChange(View arg0, boolean isFocused)
    {
      // TODO Auto-generated method stub

      /*若 ImageButton 状态为 onFocus，改变 ImageButton 的图片并改变 textView 的文字*/
      if (isFocused==true)
      {
        mTextView1.setText("图片按钮状态为:Got Focus");
        mImageButton1.setImageResource(R.drawable.iconfull);
```

```
            }
            /*若 ImageButton 状态为 offFocus，改变 ImageButton 的图片并改变 textView 的文字*/
            else
            {
              mTextView1.setText("图片按钮状态为:Lost Focus");
              mImageButton1.setImageResource(R.drawable.iconempty);
            }
          }
        });

        /*通过 onClickListener 来应答 ImageButton 的 onClick 事件*/
        mImageButton1.setOnClickListener(new OnClickListener()
        {
          public void onClick(View v)
          {
            // TODO Auto-generated method stub
            /*若 ImageButton 状态为 onClick，改变 ImageButton 的图片并改变 textView 的文字*/
            mTextView1.setText("图片按钮状态为:Got Click");
            mImageButton1.setImageResource(R.drawable.iconfull);
          }
        });

        /*通过 onClickListener 来应答 Button 的 onClick 事件*/
        mButton1.setOnClickListener(new OnClickListener()
        {
          public void onClick(View v)
          {
            // TODO Auto-generated method stub
            /*若 Button 状态为 onClick，改变 ImageButton 的图片并改变 textView 的文字*/
            mTextView1.setText("图片按钮状态为:Lost Focus");
            mImageButton1.setImageResource(R.drawable.iconempty);
          }
        });
      }
```

通过上述代码实现了图片样式的按钮效果，执行后的显示效果如图 3-3 所示，单击图片按钮后的效果如图 3-4 所示。



图 3-3　初始效果



图 3-4　单击后的效果

3.3　实现选择处理

实例 044	RadioGroup 控件实现选择处理
源码路径	光盘:\daima\044
视频路径	光盘:\视频\044
实例必备	044.响应的系统设置的事件.pdf ① Configuration 类详解 ② 重写 onConfigurationChanged 响应系统设置更改

3.3.1　实例说明

在本实例中布置了一个 TextView Widget 控件和一个 RadioGroup 控件，并在 RadioGroup 中放置 2 个 RadioButton 控件，默认样式为不选择。当程序运行后使用 onCheckedChanged 作为启动事件装置，当用户选择其中一个按钮时显示被选择项的内容，最后将 RadioButton 的选项文字显示于 TextView 当中。

3.3.2　具体实现

（1）在文件 main.xml 中分别插入 1 个 TextView 控件、1 个 RadioGroup 控件和 2 个 RadioButton 控件。

（2）使用 OnCheckedChangeListener 启动 RadioGroup 的事件，将被选中选项的文字显示在 TextView 文本框中。实例主程序文件的主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*取得 TextView、RadioGroup、RadioButton 对象*/
  mTextView1 = (TextView) findViewById(R.id.myTextView);
  mRadioGroup1 = (RadioGroup) findViewById(R.id.myRadioGroup);
  mRadio1 = (RadioButton) findViewById(R.id.myRadioButton1);
  mRadio2 = (RadioButton) findViewById(R.id.myRadioButton2);

  /*RadioGroup 用 OnCheckedChangeListener 来运行*/
  mRadioGroup1.setOnCheckedChangeListener(mChangeRadio);
}

private RadioGroup.OnCheckedChangeListener mChangeRadio = new
          RadioGroup.OnCheckedChangeListener()
{
  @Override
  public void onCheckedChanged(RadioGroup group, int checkedId)
  {
    // TODO Auto-generated method stub
```

```
        if(checkedId==mRadio1.getId())
        {
          /*把 mRadio1 的内容传到 mTextView1*/
          mTextView1.setText(mRadio1.getText());
        }
        else if(checkedId==mRadio2.getId())
        {
          /*把 mRadio2 的内容传到 mTextView1*/
          mTextView1.setText(mRadio2.getText());
        }
      }
    };
```

实例执行后的效果如图 3-5 所示，当选择一个选项后会提示已选择的值，如图 3-6 所示。

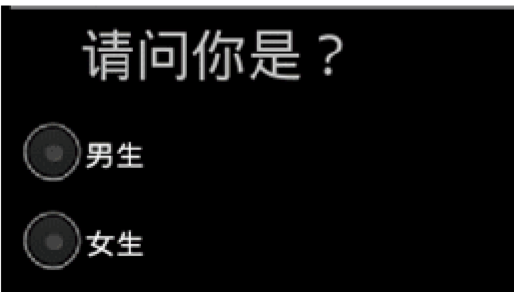


图 3-5　执行效果



图 3-6　输出提示

3.4　实现购物清单效果

实例 045	实现一个购物清单效果
源码路径	光盘:\daima\045
视频路径	光盘:\视频\045
实例必备	045.Handler 消息传递机制.pdf

3.4.1　实例说明

在 Android 系统中，CheckBox 是一个供用户选择的复选框控件。在下面的实例中，通过 CheckBox.setOnCheckedChangeListener 在程序中设置了 3 个 CheckBox 控件，分别表示 3 种物品列表，当选中其中的一个物品后，会在 TextView 控件中里显示选择的物品列表。

3.4.2　具体实现

（1）在文件 main.xml 中分别插入了 1 个 TextView 控件和 3 个 CheckBox 控件。

（2）在主程序文件中分别构建了 3 个 CheckBox 对象和 1 个 TextView 对象，然后通过响应 setOnCheckedChangeListener 事件，使用方法 onCheckedChanged()来更新 TextView 中显示的文字。在具体实现上，假定用户只有 3600 元钱，然后提供了一个货物清单供用户选择要买的商品。当用户选择商品后，会显示对应的物品，如果超出了 3600 这个额度，会显示对应的提示。主程序文件的主要代码如下所示。

```
/*声明对象变量*/
private TextView mTextView1;
private CheckBox mCheckBox1;
private CheckBox mCheckBox2;
private CheckBox mCheckBox3;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*通过 findViewById 取得 TextView 对象并调整文字内容*/
  mTextView1 = (TextView) findViewById(R.id.myTextView1);
  mTextView1.setText("你所选择的项目有: ");

  /*通过 findViewById 取得 3 个 CheckBox 对象*/
  mCheckBox1=(CheckBox)findViewById(R.id.myCheckBox1);
  mCheckBox2=(CheckBox)findViewById(R.id.myCheckBox2);
  mCheckBox3=(CheckBox)findViewById(R.id.myCheckBox3);

  /*设置 OnCheckedChangeListener 给 3 个 CheckBox 对象*/
  mCheckBox1.setOnCheckedChangeListener(mCheckBoxChanged);
  mCheckBox2.setOnCheckedChangeListener(mCheckBoxChanged);
  mCheckBox3.setOnCheckedChangeListener(mCheckBoxChanged);
}

/*声明并建构 OnCheckedChangeListener 对象*/
private CheckBox.OnCheckedChangeListener mCheckBoxChanged
= new CheckBox.OnCheckedChangeListener()
{
  /*implement onCheckedChanged 方法*/
  @Override
  public void onCheckedChanged(CompoundButton buttonView,
                               boolean isChecked)
  {
    // TODO Auto-generated method stub
    /*通过 getString()取得 CheckBox 的文字字符串*/
    String str0="所选的项目为: ";
    String str1=getString(R.string.str_checkbox1);
    String str2=getString(R.string.str_checkbox2);
    String str3=getString(R.string.str_checkbox3);
    String plus=";";
    String result="但是超过预算啰!!";
    String result2="还可以再多买几个喔!!";

    /*任一 CheckBox 被选中后，该 CheckBox 的文字会改变 TextView 的文字内容
     * 3 个对象总共 8 种情况*/
    if(mCheckBox1.isChecked()==true & mCheckBox2.isChecked()==true
        & mCheckBox3.isChecked()==true)
    {
      mTextView1.setText(str0+str1+plus+str2+plus+str3+result);
```

```
            }
            else if(mCheckBox1.isChecked()==false & mCheckBox2.isChecked()==true
                  & mCheckBox3.isChecked()==true)
            {
              mTextView1.setText(str0+str2+plus+str3+result);
            }
            else if(mCheckBox1.isChecked()==true & mCheckBox2.isChecked()==false
                  & mCheckBox3.isChecked()==true)
            {
              mTextView1.setText(str0+str1+plus+str3+result);
            }
            else if(mCheckBox1.isChecked()==true & mCheckBox2.isChecked()==true
                  & mCheckBox3.isChecked()==false)
            {
              mTextView1.setText(str0+str1+plus+str2+result);
            }
            else if(mCheckBox1.isChecked()==false & mCheckBox2.isChecked()==false
                  & mCheckBox3.isChecked()==true)
            {
              mTextView1.setText(str0+str3+plus+result2);
            }
            else if(mCheckBox1.isChecked()==false & mCheckBox2.isChecked()==true
                  & mCheckBox3.isChecked()==false)
            {
              mTextView1.setText(str0+str2);
            }
            else if(mCheckBox1.isChecked()==true & mCheckBox2.isChecked()==false
                  & mCheckBox3.isChecked()==false)
            {
              mTextView1.setText(str0+str1);
            }
            else if(mCheckBox1.isChecked()==false & mCheckBox2.isChecked()==false
                  & mCheckBox3.isChecked()==false)
            {
              mTextView1.setText(str0);
            }
         }
      };
}
```

实例执行后的效果如图 3-7 所示，当选择一种商品后会在下方显示对应的提示信息，如图 3-8 所示。如果超出了预算价值 3600，则会显示超出预算的提示，如图 3-9 所示。



图 3-7　执行效果



图 3-8　显示提示效果



图 3-9　超出预算提示效果

3.5　更换图片的相框

实例 046	为图片实现相框效果
源码路径	光盘:\daima\046
视频路径	光盘:\视频\046
实例必备	046.Activity 基础.pdf ① Activity 的状态及状态间的转换 ② Activity 栈 ③ Activity 的生命周期

3.5.1　实例说明

在本实例中，需要预先准备 3 张图片，其中 2 张是外框图，1 张是内框图，将这 3 张图片放在 res\drawable 目录下面。图片是 PNG 格式的图形文件，图像大小是手机屏幕大小，读者可以依据手机的分辨率来调整 ImageView 的大小。

在 Layout 布局中创建了两个 ImageView 图像视图，并且以绝对坐标的方式“堆”在一起，然后在其下方放置两个 Button 控件，单击按钮后能够实现切换图片功能，在此需要设置 Button 事件中处理置换图片的动作。

当单击 Button1 后，在 ImageView1 中会显示 right 中的图片；单击 Button2 后，在 ImageView1 中会显示 left 的图片，而 ImageView2 都是固定不动的图片。

3.5.2　具体实现

（1）在文件 main.xml 中分别创建两个 ImageView 控件，其中一个作为外框，另一个作为内框。在摆放图片时，需要做一个排序堆栈顺序，将前景图放在上方（以 AbsoluteLayout），将背景图放在前景图的下方。

（2）编写主程序文件，其核心功能是通过 getResources()方法实现的，此方法负责访问 Resource ID，无论是访问资源中的图文件，还是文字，都要用到 getResources()。在此使用 getResources().getDrawable()载入 res\drawable 中的图文件，并将图片放置在 ImageView 当中。主程序文件的主要代码如下所示。

```
/*声明 Button、ImageView 对象*/
private ImageView mImageView01;
private ImageView mImageView02;
private Button mButton01;
private Button mButton02;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
```

```
/*取得 Button、ImageView 对象*/
mImageView01 = (ImageView)findViewById(R.id.myImageView1);
mImageView02 = (ImageView)findViewById(R.id.myImageView2);
mButton01 = (Button) findViewById(R.id.myButton1);
mButton02 = (Button) findViewById(R.id.myButton2);

/*设置 ImageView 背景图*/
mImageView01.setImageDrawable(getResources().
                getDrawable(R.drawable.right));
mImageView02.setImageDrawable(getResources().
                getDrawable(R.drawable.aaa));

/*用 OnClickListener 事件启动*/
mButton01.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
 {
  /*当启动后，ImageView 立刻换背景图*/
  mImageView01.setImageDrawable(getResources().
                getDrawable(R.drawable.right));
 }
});

mButton02.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
 {
  mImageView01.setImageDrawable(getResources().
                getDrawable(R.drawable.left));
 }
});

}
```

实例执行后的初始效果如图 3-10 所示，当分别单击 pic1 和 pic2 按钮后，会分别显示用 Left 和 Right 素材的相框，如图 3-11 所示。



图 3-10　初始效果

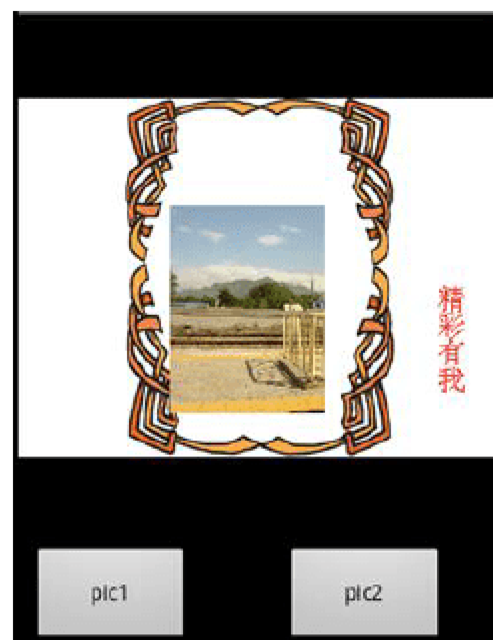


图 3-11　不同素材相框

对于上述实例来说，读者可以将两个 ImageButton Widget 堆栈在一起，这样不但有背景图，而且还有按钮事件，具体代码如下所示。

```
<ImageButton
  android:id="@+id/myImageButton1"
  android:state_focused="true"
  android:layout_width="320px"
  android:layout_height="280px"
  android:layout_x="0px"
  android:layout_y="36px"
/>
```

使用 ImageButton 的方法比较简单，而堆栈的技巧可参考该范例程序，不同之处就是只要单击图片，即可直接换图，不需要再单击 Button 做更换。需要注意的是，图片大小要作调整，否则可能与 ImageButton 不合。

另外，对于 res\drawable 目录下的素材图片，图片名字不能是纯数字组成的，否则程序将会出现错误。

3.6　选择自己喜欢的球队

实例 047	使用 Spinner 实现选择处理
源码路径	光盘:\daima\047
视频路径	光盘:\视频\047
实例必备	047.操作 Activity.pdf ① 使用 LauncherActivity 类 ② 使用 ExpandableListActivity 类 ③ 使用 PreferenceActivity 和 PreferenceFragment

3.6.1　实例说明

Spinner 是一个下拉菜单，类似于 Swing 中的 Combo Box、HTML 中的 <select>。因为手机画面有限，所以要在有限的范围内选择项目，下拉菜单是唯一、也是较好的选择。在本实例中自定义了一个下拉菜单样式，然后调用方法 setDropDownViewResource()以 XML 的方式定义下拉菜单要显示的模样。在实例中除了自定义下拉菜单外，还编程实现了一段动画效果，增加了实例的美观性。

3.6.2　具体实现

（1）在文件 main.xml 中分别插入 1 个 TextView 控件和 1 个 Spinner 控件。

（2）编写文件 myspinner.xml 用于布局弹出下拉菜单界面的样式，在其中使用了 TextView 组件。

（3）编写动画样式文件 my_anim.xml ，Android 的动画由 4 种类型（type）组成，分别是 alpha、scale、translate 和 rotate，在下面的自定义动画代码中使用了 translate 和 alpha，具体代码如下。

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <translate
```

```
  android:fromXDelta="0"
  android:toXDelta="-100%p"
  android:duration="300"
  >
  </translate>
  <alpha
  android:fromAlpha="1.0"
  android:toAlpha="0.0"
  android:duration="300">
  </alpha>
</set>
```

（4）编写文件 example055.java，在新建 ArrayAdapter 时使用 ArrayAdapter(Context context, int textViewResourceId, T[ ] objects)生成器，参数 textViewResourceId 使用 Android 提供的 ResourceID，参数 objects 为必须传递的字符串数组（String Array）。文件 example055.java 的主要代码如下所示。

```
public class example055 extends Activity
{
  private static final String[ ] countriesStr =
  { "曼联", "利物浦", "切尔西", "阿森纳" };
  private TextView myTextView;
  private Spinner mySpinner;
  private ArrayAdapter<String> adapter;
  Animation myAnimation;

  /** Called when the activity is first created.*/
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /*载入 main.xml Layout*/
    setContentView(R.layout.main);

    /*以 findViewById()取得对象*/
    myTextView = (TextView) findViewById(R.id.myTextView);
    mySpinner = (Spinner) findViewById(R.id.mySpinner);

    adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item, countriesStr);
    /*myspinner_dropdown 为自定义下拉菜单模式，定义在 res\layout 目录下*/
    adapter.setDropDownViewResource(R.layout.myspinner_dropdown);

    /*将 ArrayAdapter 添加到 Spinner 对象中*/
    mySpinner.setAdapter(adapter);

    /*将 mySpinner 添加到 OnItemSelectedListener 中*/
    mySpinner.setOnItemSelectedListener
     (new Spinner.OnItemSelectedListener()
    {
      @Override
      public void onItemSelected
       (AdapterView<?> arg0, View arg1, int arg2,
```

```
            long arg3)
        {
            /*将所选 mySpinner 的值带入 myTextView 中*/
            myTextView.setText("您选择的是" + countriesStr[arg2]);
            /*显示 mySpinner*/
            arg0.setVisibility(View.VISIBLE);
        }

        @Override
        public void onNothingSelected(AdapterView<?> arg0)
        {
            // TODO Auto-generated method stub
        }
    });

    /*取得 Animation 定义在 res\anim 目录下*/
    myAnimation = AnimationUtils.loadAnimation(this, R.anim.my_anim);

    /*将 mySpinner 添加到 OnTouchListener 中*/
    mySpinner.setOnTouchListener(new Spinner.OnTouchListener()
    {

        @Override
        public boolean onTouch(View v, MotionEvent event)
        {
            /*将在 mySpinner 区域运行 Animation 动画效果*/
            v.startAnimation(myAnimation);
            /*将 mySpinner 隐藏*/
            v.setVisibility(View.INVISIBLE);
            return false;
        }

    });

    mySpinner.setOnFocusChangeListener(new Spinner.OnFocusChangeListener()
    {
        @Override
        public void onFocusChange(View v, boolean hasFocus)
        {
            // TODO Auto-generated method stub
        }
    });
  }
}
```

执行后的效果如图 3-12 所示，单击下拉菜单后显示悬浮选项效果界面，在其中有 4 个选项供用户选择，如图 3-13 所示。

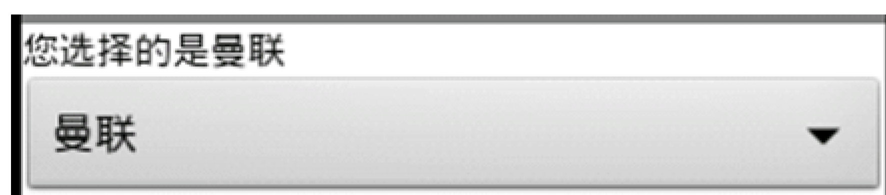


图 3-12　执行效果

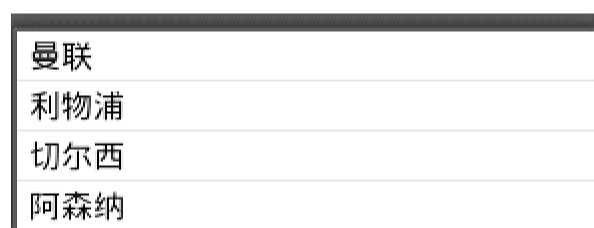


图 3-13　4 个选项

当选择一个选项后，会显示出对应的提示信息，例如，选择了“阿森纳”后的效果如图 3-14 所示。

在上述实例中，使用 Adapter 的 setDropDownViewResource 可以设置下拉菜单的显示样式，设置样式的 XML 文件被保存在 res\layout 目录下。可以针对下拉菜单中的 TextView 进行设置，如同本程序中的 R.layout.myspinner_dropdown 即为自定义的下拉菜单 TextView 样式。除了改变下拉菜单样式外，也对 Spinner 做了一些动态效果，单击 Spinner 时，晃动 Spinner 即可出现下拉菜单（myAnimation）。

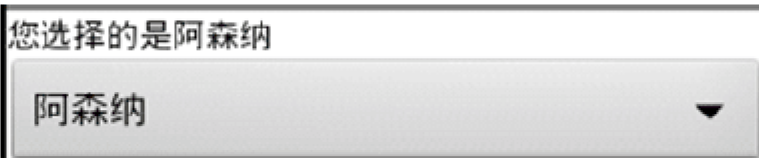


图 3-14　提示信息

3.7　实现文件上传功能

实例 048	在手机中实现文件上传功能
源码路径	光盘:\daima\048
视频路径	光盘:\视频\048
实例必备	048.配置 Activity.pdf

3.7.1　实例说明

在开发 Android 应用程序的过程中，采用 POST 方式向服务器传递数据的基本步骤如下所示。

（1）利用 Map 集合对数据进行获取并进行数据处理，例如：

```
if (params!=null&&!params.isEmpty()) {
  for (Map.Entry<String, String> entry:params.entrySet()) {
    sb.append(entry.getKey()).append("=");
    sb.append(URLEncoder.encode(entry.getValue(),encoding));
    sb.append("&");
      }
    sb.deleteCharAt(sb.length()-1);
    }
```

（2）新建一个 StringBuilder 对象，得到 POST 传给服务器的数据，例如：

```
sb=new StringBuilder()
byte[ ] data=sb.toString().getBytes();
```

（3）新建一个 HttpURLConnection 的 URL 对象，打开连接并传递服务器的 path，例如：

```
connection=(HttpURLConnection) new URL(path).openConnection();
```

（4）设置超时和允许对外连接数据，例如：

```
connection.setDoOutput(true);
```

（5）设置连接的 setRequestProperty 属性，例如：

```
connection.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
connection.setRequestProperty("Content-Length", data.length+"");
```

（6）得到连接输出流，例如：

```
outputStream =connection.getOutputStream();
```

（7）把得到的数据写入输出流中并刷新，例如：

```
outputStream.write(data);
outputStream.flush();
```

3.7.2　具体实现

（1）打开 Eclipse，新建一个名为 ServerForPOSTMethod 的 Web 工程，并自动生成配置文件 web.xml。

（2）创建一个名为 ServletForPOSTMethod 的 Servlet，功能是接收并处理通过 POST 方式上传的数据。实现文件 ServletForPOSTMethod.java 的具体代码如下所示。

```
@WebServlet("/ServletForPOSTMethod")
public class ServletForPOSTMethod extends HttpServlet {
	private static final long serialVersionUID = 1L;
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Servlet
Exception, IOException {
		String name= request.getParameter("name");
		String age= request.getParameter("age");
		System.out.println("name from POST method: " + name );
		System.out.println("age from POST method: " + age );
	}
}
```

（3）在配置文件 web.xml 中配置 ServletForGETMethod，具体实现代码如下所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"   xsi:schemaLocation="http://java.sun.com/xml/ns/
javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>ServerForPOSTMethod</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

（4）打开 Eclipse，新建一个名为 POST 的 Android 工程。然后编写界面布局文件 main.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/title"
    />
    <EditText
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:id="@+id/title"
```

```
    />
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/length"
    />
    <EditText
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:numeric="integer"
     android:id="@+id/length"
    />
    <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button"
    android:onClick="save"
    />
</LinearLayout>
```

（5）编写文件 UploadUserInformationByPOSTActivity.java，具体实现代码如下所示。

```
public class UploadUserInformationByPOSTActivity extends Activity {
    private EditText titleText;
    private EditText lengthText;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        titleText = (EditText) this.findViewById(R.id.title);
        lengthText = (EditText) this.findViewById(R.id.length);
    }

    public void save(View v){
     String title = titleText.getText().toString();
     String length = lengthText.getText().toString();
     try {
            boolean result = false;

            result = UploadUserInformationByPostService.save(title, length);

            if(result){
                        Toast.makeText(this, R.string.success, 1).show();
                  }else{
                        Toast.makeText(this, R.string.fail, 1).show();
                  }
            } catch (Exception e) {
                  e.printStackTrace();
                  Toast.makeText(this, R.string.fail, 1).show();
            }
    }
}
}
```

（6）编写业务类的实现文件 UploadUserInformationByPostService.java，主要实现代码如下所示。

```
public class UploadUserInformationByPostService {
	public static boolean save(String title, String length) throws Exception{
		String path = "http://192.168.1.100:8080/ServerForPOSTMethod/ServletForPOSTMethod";
		Map<String, String> params = new HashMap<String, String>();
		params.put("name", title);
		params.put("age", length);
		return sendPOSTRequest(path, params, "UTF-8");
	}

	/**
	 * 发送 POST 请求
	 * @param path 请求路径
	 * @param params 请求参数
	 * @return
	 */
	private static boolean sendPOSTRequest(String path, Map<String, String> params, String encoding)
throws Exception{
		//title=liming&length=30
		StringBuilder sb = new StringBuilder();
		if(params!=null && !params.isEmpty()){
			for(Map.Entry<String, String> entry : params.entrySet()){
				sb.append(entry.getKey()).append("=");
				sb.append(URLEncoder.encode(entry.getValue(), encoding));
				sb.append("&");
			}
			sb.deleteCharAt(sb.length() - 1);
		}
		byte[ ] data = sb.toString().getBytes();

		HttpURLConnection conn = (HttpURLConnection) new URL(path).openConnection();
		conn.setConnectTimeout(5000);
		conn.setRequestMethod("POST");
		conn.setDoOutput(true);                    //允许对外传输数据
		conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
		conn.setRequestProperty("Content-Length", data.length+"");
		OutputStream outStream = conn.getOutputStream();
		outStream.write(data);
		outStream.flush();
		if(conn.getResponseCode() == 200){
			return true;
		}
		return false;
	}
}
```

（7）编写配置文件 AndroidManifest.xml，声明网络访问权限，主要代码如下所示。

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.guan.internet.userInformation.post"
    android:versionCode="1"
    android:versionName="1.0" >
```

```
    <uses-sdk android:minSdkVersion="8" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name="com.guan.internet.userInformation.post.UploadUserInformationByPOSTActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
<uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

至此，整个实例讲解完毕，执行后的效果如图 3-15 所示。输入用户名和年龄后，单击 save 按钮，输入的数据将会上传至服务器。

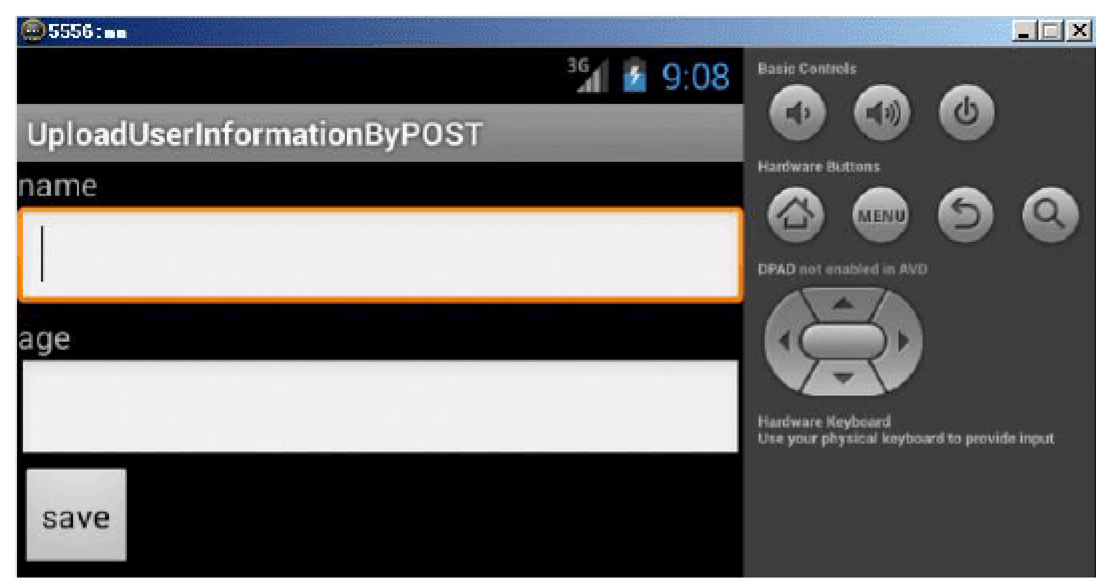


图 3-15　执行效果

3.8　日期和时间选择器

实例 049	在屏幕中实现日期和时间选择器
源码路径	光盘:\daima\049
视频路径	光盘:\视频\049
实例必备	049.启动、关闭 Activity.pdf

3.8.1　实例说明

在 Android API 中有两个十分重要的对象，即 DatePicker 和 TimePicker，通过这两个对象可以实现动态输入日期与时间的功能。本实例中使用了 DatePicker、TimePicker 和 TextView 这 3 个对象，TextView 用于显示日期与时间，默认带入目前系统的日期与时间，DatePicker 与 TimePicker 用于让用户动态调整日期与时间。当用户调整了 DatePicker 日期或 TimePicker 时间时，在 TextView 中显示的日期与时间也会随之改变。

3.8.2　具体实现

编写主程序文件，使用 updateDisplay()方法来设置在 TextView 中所显示的日期时间，使用 java.util.Calendar 对象来获取当前系统的时间，并将该时间传入 TextView 中。如果用户改变了 DatePicker 中的年、月、日，会运行如下两类操作。

（1）触发 DatePicker 的 onDateChange 事件，并同时运行方法 updateDisplay()重新设置在 TextView 中显示的日期。

（2）触发 TimePicker 的 onTimeChange 事件，运行方法 updateDisplay()重新设置在 TextView 中显示的时间。

主程序文件的主要代码如下所示。

```
/*声明日期及时间变量*/
private int mYear;
private int mMonth;
private int mDay;
private int mHour;
private int mMinute;
/*声明对象变量*/
TextView tv;
TimePicker tp;
DatePicker dp;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  /*取得目前日期与时间*/
  Calendar c=Calendar.getInstance();
  mYear=c.get(Calendar.YEAR);
  mMonth=c.get(Calendar.MONTH);
  mDay=c.get(Calendar.DAY_OF_MONTH);
  mHour=c.get(Calendar.HOUR_OF_DAY);
  mMinute=c.get(Calendar.MINUTE);

  super.onCreate(savedInstanceState);
  /*载入 main.xml Layout*/
  setContentView(R.layout.main);

  /*取得 TextView 对象，并调用 updateDisplay()来设置显示的初始日期时间*/
  tv= (TextView) findViewById(R.id.showTime);
  updateDisplay();

  /*取得 DatePicker 对象，以 init()设置初始值与 onDateChangeListener()*/
  dp=(DatePicker)findViewById(R.id.dPicker);
  dp.init(mYear,mMonth,mDay,new DatePicker.OnDateChangedListener()
  {
    @Override
    public void onDateChanged(DatePicker view,int year,
```

```
                                        int monthOfYear,int dayOfMonth)
          {
            mYear=year;
            mMonth= monthOfYear;
            mDay=dayOfMonth;
            /*调用 updateDisplay()来改变显示日期*/
            updateDisplay();
          }
        });
        /*取得 TimePicker 对象，并设置为 24 小时制显示*/
        tp=(TimePicker)findViewById(R.id.tPicker);
        tp.setIs24HourView(true);

        /*setOnTimeChangedListener，并覆盖 onTimeChanged 事件*/
        tp.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener()
        {
          @Override
          public void onTimeChanged(TimePicker view,
                                    int hourOfDay,
                                    int minute)
          {
            mHour=hourOfDay;
            mMinute=minute;
            /*调用 updateDisplay()来改变显示时间*/
            updateDisplay();
          }
        });
      }

      /*设置显示日期时间的方法*/
      private void updateDisplay()
      {
        tv.setText(
          new StringBuilder().append(mYear).append(" / ")
                             .append(format(mMonth + 1)).append(" / ")
                             .append(format(mDay)).append("   ")
                             .append(format(mHour)).append(":")
                             .append(format(mMinute))
        );
      }

      /*日期时间显示两位数的方法*/
      private String format(int x)
      {
        String s=""+x;
        if(s.length()==1) s="0"+s;
        return s;
      }
    }
```

至此，整个实例介绍完毕，执行后会显示一个数字时钟效果。用户可以分别单击＋和－来自动选

择日期和时间，具体效果如图 3-16 所示。

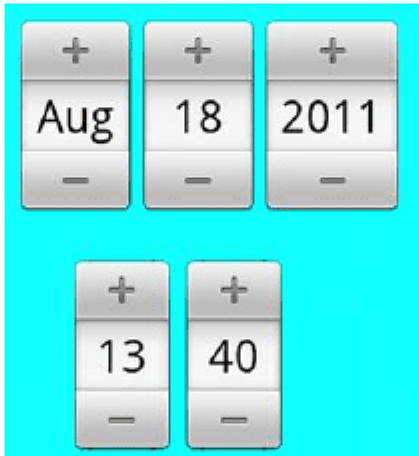


图 3-16　执行效果

3.9　动态排版屏幕布局

实例 050	动态排版手机屏幕
源码路径	光盘:\daima\050
视频路径	光盘:\视频\050
实例必备	050.Activity 数据交换.pdf

3.9.1　实例说明

GridView 能够实现网格化的二维排版。在本实例中，将联合使用 GridView 和 ArrayAdapter 实现动态排版功能。首先在屏幕中插入 2 个按钮作为动态放入 GridView 的开关，第一个按钮设置屏幕显示为两行两列样式，在里面放入 4 个 Item；另一个按钮设置屏幕显示为 3 行 3 列显示样式，在其中放入 9 个 Item，这样就实现了对文字的动态排版处理。

3.9.2　具体实现

编写主程序文件，在其中设置了 mButton01 和 mButton02 两个按钮对象，并分别为其设置了 OnClickListener 单击监听事件，在按钮单击事件中会处理配置 GridView 的方法。为了方便地配置 GridView，通过 setNumColumns()方法设置了在 GridView 中将要显示的字段数量，然后初始化了 Array Adapter 对象 aryAdapter1，这样就可以使用调用 GridView.setAdapter()的方式，将 String 类型的 Item 放入 GridView 中。

主程序文件的主要实现代码如下：

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*4 个字符串数组*/
  mGames1 = new String[ ]
                {
```

```
                        getResources().getString(R.string.str_list1),
                        getResources().getString(R.string.str_list2),
                        getResources().getString(R.string.str_list3),
                        getResources().getString(R.string.str_list4)
                    };

/*9 个字符串数组*/
mGames2 = new String[ ]
                    {
                        getResources().getString(R.string.str_list1),
                        getResources().getString(R.string.str_list2),
                        getResources().getString(R.string.str_list3),
                        getResources().getString(R.string.str_list4),
                        getResources().getString(R.string.str_list1),
                        getResources().getString(R.string.str_list2),
                        getResources().getString(R.string.str_list3),
                        getResources().getString(R.string.str_list4),
                        getResources().getString(R.string.str_list1)
                    };

mButton01 = (Button)findViewById(R.id.myButton1);
mButton02 = (Button)findViewById(R.id.myButton2);
mGridView01 = (GridView)findViewById(R.id.myGridView1);

mTextView01 = (TextView)findViewById(R.id.myTextView1);

mButton01.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    // TODO Auto-generated method stub

    /*4 个元素，以 2 列方式呈现(2×2)*/
    mGridView01.setNumColumns(2);

    aryAdapter1 = new ArrayAdapter<String>
    (example62.this, R.layout.simple_list_item_1_small, mGames1);

    mGridView01.setAdapter(aryAdapter1);
    //mGridView01.setScrollBarStyle(DEFAULT_KEYS_DIALER);
    mGridView01.setSelection(2);
    mGridView01.refreshDrawableState();
  }
});

mButton02.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
```

```
        {
          // TODO Auto-generated method stub

          /*9 个元素，以 3 列方式呈现(3×3)*/
          mGridView01.setNumColumns(3);

          aryAdapter1 = new ArrayAdapter<String>
          (example16.this, R.layout.simple_list_item_1_small, mGames2);

          mGridView01.setAdapter(aryAdapter1);
        }
      });

      mGridView01.setOnItemClickListener
                    (new GridView.OnItemClickListener()
      {
        @Override
        public void onItemClick(AdapterView<?> parent,
                                View v, int position, long arg3)
        {
          // TODO Auto-generated method stub

          /*判断 Adapter 里的元素个数，判断被单击的是第几个元素名称*/
          switch(aryAdapter1.getCount())
          {
            case 4:
              /*position 为 GridView 里的元素索引值*/
              mTextView01.setText(mGames1[position]);
              break;
            case 9:
              mTextView01.setText(mGames2[position]);
              break;
          }
        }
      });
    }
```

执行后先显示有两个按钮的界面，当单击“显示（2×2）个”按钮后会显示两行两列的排列样式，如图 3-17 所示；当单击“显示（3×3）个”按钮后会显示 3 行 3 列的排列样式，如图 3-18 所示。

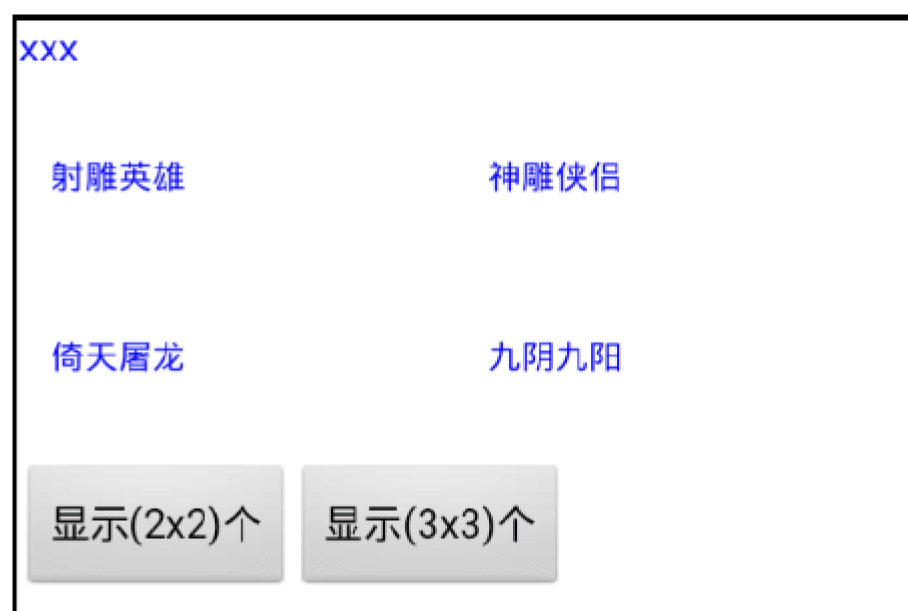


图 3-17　2×2 排列



图 3-18　3×3 排列

3.10　加载手机磁盘中的文件

实例 051	加载手机磁盘中的文件
源码路径	光盘:\daima\051
视频路径	光盘:\视频\051
实例必备	051.Activity 的加载模式.pdf ① standard 加载模式 ② singleTop 加载模式 ③ singleTask 加载模式 ④ singleInstance 加载模式

3.10.1　实例说明

在 Android 项目程序中，通常将图片文件保存在 res\drawable 目录下，这样可以通过 R.drawable.id 来获取图片的 id，可以在程序中任意调用这幅图片。但是如果没有将此图片保存在 res\drawable 目录下，例如，仅仅想从存储卡中获取一幅图片，此时需要用 Android API 中的 Bitmap 对象来实现。

在本实例中，将使用 decodeFile()方法来加载手机磁盘中的图片文件，并显示在手机屏幕中。

3.10.2　具体实现

（1）编写文件 main.xml，在屏幕中分别插入 1 个 TextView 控件、1 个 ImageView 控件和 1 个 Button 控件。

（2）编写主程序文件，使用 decodeFile(fileName)方法获取 Bitmap 对象，然后使用 setImageBitmap()来设置此 Bitmap 对象的显示。主程序文件的主要代码如下所示。

```
/*声明对象变量*/
private ImageView mImageView;
private Button mButton;
private TextView mTextView;
private String fileName="/data/123.png";

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  /*载入 main.xml Layout*/
  setContentView(R.layout.main);

  /*取得 Button 对象，并添加 onClickListener*/
  mButton = (Button)findViewById(R.id.mButton);
  mButton.setOnClickListener(new Button.OnClickListener()
  {
```

```
    public void onClick(View v)
    {
    /*取得对象*/
      mImageView = (ImageView)findViewById(R.id.mImageView);
      mTextView=(TextView)findViewById(R.id.mTextView);
      /*检查文件是否存在*/
      File f=new File(fileName);
      if(f.exists())
      {
        /*产生 Bitmap 对象，并放入 mImageView 中*/
        Bitmap bm = BitmapFactory.decodeFile(fileName);
        mImageView.setImageBitmap(bm);
        mTextView.setText(fileName);
      }
      else
      {
        mTextView.setText("文件不存在");
      }
    }
  });
}
```

执行后将加载指定的图片并显示在屏幕中，如图 3-19 所示。

其实本实例应该属于图片处理的范畴，放在本章的目的是讲解 decodeFile()方法和 Button 控件联合使用的效果。在放置图片时需要将素材图片放在 res 目录下，在此初学者经常遇到如下问题。

通过 BitmapFactory.decodeFile(pathName)加载一个 480×320 的图片，但是当使用时，通过 getWidth()和 getHeight()取得的 Bitmap 大小却只有 320×213。本来应该全屏显示，结果只是显示在了屏幕的左上方。

这是因为把素材图片放在了 drawable-hdpi 目录下，如果把图片放在 drawable-ldpi 和 drawable-mdpi 中就不会有问题了。通常在 res 目录下有 3 个用于保存素材图片的文件夹，如图 3-20 所示。



图 3-19　执行效果



图 3-20　保存素材图片的文件夹

☑　drawable-hdpi：存放高分辨率的图片，如 WVGA（480×800）、FWVGA（480×854）。
☑　drawable-mdpi：存放中等分辨率的图片，如 HVGA（320×480）。
☑　drawable-ldpi：存放低分辨率的图片，如 QVGA（240×320）。

3.11 动态添加/删除 Spinner 菜单

实例 052	动态添加/删除 Spinner 菜单
源码路径	光盘:\daima\052
视频路径	光盘:\视频\052
实例必备	052.使用 Fragment.pdf ① Fragment 基础 ② 创建 Fragment

3.11.1 实例说明

在本书前面的实例中已经讲解过用 Spinner 自定义菜单和实现事件交互的方法。本实例的功能将更加高级，在本实例中将利用 ArrayList（动态数组）的依赖性动态增减 Spinner 下拉菜单中的选项。

在本实例中将设计一个 EditText 输入框，在用户输入了文字并单击“添加”按钮的同时，会将输入的值添加至 Spinner 下拉菜单的最后一项，接着 Spinner 会停留在刚添加的选项上；单击“删除”按钮则会删除选择的 Spinner 选项。

3.11.2 具体实现

（1）编写文件 main.xml，在其中插入 1 个 TextView 控件、1 个 Button 控件和 1 个 EditText 控件。

（2）编写文件 add.java，主要代码如下所示。

```
public class add extends Activity
{
  private static final String[ ] countriesStr =
  { "Android 多媒体", "Android 网络", "Android 驱动", "Android 游戏" };
  private TextView myTextView;
  private EditText myEditText;
  private Button myButton_add;
  private Button myButton_remove;
  private Spinner mySpinner;
  private ArrayAdapter<String> adapter;
  private List<String> allCountries;
  /** Called when the activity is first created.*/
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /*载入 main.xml Layout*/
    setContentView(R.layout.main);
    allCountries = new ArrayList<String>();
    for (int i = 0; i < countriesStr.length; i++)
    {
```

```
    allCountries.add(countriesStr[i]);
}
/*new ArrayAdapter 对象并将 allCountries 传入*/
adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item, allCountries);
adapter
        .setDropDownViewResource
         (android.R.layout.simple_spinner_dropdown_item);
/*用 findViewById()取得对象*/
myTextView = (TextView) findViewById(R.id.myTextView);
myEditText = (EditText) findViewById(R.id.myEditText);
myButton_add = (Button) findViewById(R.id.myButton_add);
myButton_remove = (Button) findViewById(R.id.myButton_remove);
mySpinner = (Spinner) findViewById(R.id.mySpinner);
/*将 ArrayAdapter 添加到 Spinner 对象中*/
mySpinner.setAdapter(adapter);
/*将 myButton_add 添加到 OnClickListener 中*/
myButton_add.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        String newCountry = myEditText.getText().toString();
        /*先比较添加的值是否已存在，不存在才可添加*/
        for (int i = 0; i < adapter.getCount(); i++)
        {
            if (newCountry.equals(adapter.getItem(i)))
            {
                return;
            }
        }
        if (!newCountry.equals(""))
        {
            /*将值添加到 adapter 中*/
            adapter.add(newCountry);
            /*取得添加的值的位置*/
            int position = adapter.getPosition(newCountry);
            /*将 Spinner 选择在添加的值的位置*/
            mySpinner.setSelection(position);
            /*将 myEditText 清空*/
            myEditText.setText("");
        }
    }
});
/*将 myButton_remove 添加到 OnClickListener 中*/
myButton_remove.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        if (mySpinner.getSelectedItem() != null)
```

```
            {
                /*删除 mySpinner 的值*/
                adapter.remove(mySpinner.getSelectedItem().toString());
                /*将 myEditText 清空*/
                myEditText.setText("");
                if (adapter.getCount() == 0)
                {
                    /*将 myTextView 清空*/
                    myTextView.setText("");
                }
            }
        }
    });
    /*将 mySpinner 添加到 OnItemSelectedListener 中*/
    mySpinner.setOnItemSelectedListener
    (new Spinner.OnItemSelectedListener()
    {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2,
                long arg3)
        {
            /*将所选 mySpinner 的值带入 myTextView 中*/
            myTextView.setText(arg0.getSelectedItem().toString());
        }
        @Override
        public void onNothingSelected(AdapterView<?> arg0)
        {
        }
    });
  }
}
```

上述代码的实现流程如下所示。

① 定义数组 String[ ]，保存 4 个数据。

② 载入 main.xml 的布局，然后新建 ArrayAdapter 对象并将 allCountries 传入。

③ 用 findViewById()取得对象，并将 ArrayAdapter 添加到 Spinner 对象中。

④ 实现添加处理：先比较添加的值是否已存在，不存在才可添加；然后将值添加到 adapter 中，并取得添加的值的位置，将 Spinner 选择在添加的值的位置；最后，将 myEditText 清空。

⑤ 通过 setOnClickListener 将 myButton_remove 添加到 OnClickListener 中：首先，删除 mySpinner 的值；然后，将 myEditText 清空；最后，将 myTextView 清空。

⑥ 使用 setOnItemSelectedListener 监听用户选择选项，将选择的项 mySpinner 添加到 OnItemSelectedListener 中，并将所选项 mySpinner 的值传入 myTextView 中。

执行后的效果如图 3-21 所示；在文本框中输入一个选项并单击“添加”按钮后，会将输入的信息添加到下拉列表框中，如图 3-22 所示；当单击“删除”按钮后会删除当前下拉列表框中显示的选项信息。

在本实例中，setDropDownViewResource 的功能是设置用户单击 Spinner 后出现的下拉菜单样式，除了可以使用自定义方式改变 TextView 内容外，Android 中还提供了如下两种基本的样式。



图 3-21　执行效果



图 3-22　添加一个值

☑　android.R.layout.simple_spinner_item：设置 TextView 的下拉菜单样式。
☑　android.R.layout.simple_spinner_dropdown_item：除了有 TextView 外，在右边还显示了 Radio 的下拉菜单。

（3）在界面中生成此自定义控件类对象并加以使用。

3.12　使用 OptionsMenu 在屏幕中自定义菜单

实例 053	使用 OptionsMenu 在屏幕中自定义菜单
源码路径	光盘:\daima\053
视频路径	光盘:\视频\053
实例必备	053.Intent 和 IntentFilter 基础.pdf ① Intent 启动不同组件的方法 ② Intent 的构成 ③ Intent 的基本用法

3.12.1　实例说明

菜单是用户界面中最常见的、使用非常频繁的元素之一，Android 中的菜单被分为 3 种，分别是选项菜单（OptionsMenu）、上下文菜单（ContextMenu）和子菜单（SubMenu），在本实例中使用了 OptionsMenu。菜单 OptionsMenu 中存在如下 5 个方法。

☑　public boolean onCreateOptionsMenu(Menu menu)：使用此方法调用 OptionsMenu。
☑　public boolean onOptionsItemSelected(MenuItem item)：选中菜单项后发生的动作。
☑　public void onOptionsMenuClosed(Menu menu)：菜单关闭后发生的动作。
☑　public boolean onPrepareOptionsMenu(Menu menu)：选项菜单显示之前 onPrepareOptionsMenu 方法会被调用，可以用此方法来根据当时的情况调整菜单。
☑　public boolean onMenuOpened(int featureId, Menu menu)：打开菜单后发生的动作。

3.12.2　具体实现

1．设置默认样式

默认样式是指在屏幕底部弹出的菜单，此菜单通常被称为选项菜单 OptionsMenu。在一般情况下，

选项菜单最多可以显示 2×3 的菜单项，因为在这些菜单项中有文字和图标，所以也被称为 Icon Menus。如果多于 6 项，从第 6 项开始会自动被隐藏，并在第 6 项中出现一个 More 图标，单击 More 图标后才出现第 6 项及以后的菜单项，这些菜单项通常被称为 Expanded Menus。

2．编写重载方法

编写重载方法 onCreateOptionsMenu(Menu menu)，在此方法中添加菜单项并返回 true，如果返回 false 表示不会显示菜单。文件 DefaultMenu.java 中的主要代码如下所示。

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

menu.add(Menu.NONE, Menu.FIRST + 1, 5, "删除").setIcon(
android.R.drawable.ic_menu_delete);
menu.add(Menu.NONE, Menu.FIRST + 2, 2, "保存").setIcon(
android.R.drawable.ic_menu_edit);
menu.add(Menu.NONE, Menu.FIRST + 3, 6, "帮助").setIcon(
android.R.drawable.ic_menu_help);
menu.add(Menu.NONE, Menu.FIRST + 4, 1, "添加").setIcon(
android.R.drawable.ic_menu_add);
menu.add(Menu.NONE, Menu.FIRST + 5, 4, "详细").setIcon(
android.R.drawable.ic_menu_info_details);
menu.add(Menu.NONE, Menu.FIRST + 6, 3, "发送").setIcon(
android.R.drawable.ic_menu_send);
return true;
}
```

在上述代码中，使用 setIcon()方法为菜单设置图标，这里使用的是系统自带的图标，以 android.R 开头的资源是系统提供的，个人提供的资源是以 R 开头。另外，在 add()方法中有如下 4 个参数。

☑ 组别：如果不分组，就写 Menu.NONE。

☑ id：Android 根据此 id 确定不同的菜单。

☑ 顺序：哪个菜单现在在前面由这个参数的大小决定。

☑ 文本：菜单的显示文本。

3．为菜单项注册事件

在文件 DefaultMenu.java 中使用 onOptionsItemSelected(MenuItem item)方法为菜单项注册事件，主要代码如下所示。

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case Menu.FIRST + 1:
Toast.makeText(this, "删除菜单被点击了", Toast.LENGTH_LONG).show();
break;
case Menu.FIRST + 2:
Toast.makeText(this, "保存菜单被点击了", Toast.LENGTH_LONG).show();
break;
case Menu.FIRST + 3:
Toast.makeText(this, "帮助菜单被点击了", Toast.LENGTH_LONG).show();
break;
```

```
case Menu.FIRST + 4:
Toast.makeText(this, "添加菜单被点击了", Toast.LENGTH_LONG).show();
break;
case Menu.FIRST + 5:
Toast.makeText(this, "详细菜单被点击了", Toast.LENGTH_LONG).show();
break;
case Menu.FIRST + 6:
Toast.makeText(this, "发送菜单被点击了", Toast.LENGTH_LONG).show();
break;
}
return false;
}
```

至此，此实例实现完毕，执行后的效果如图 3-23 所示。

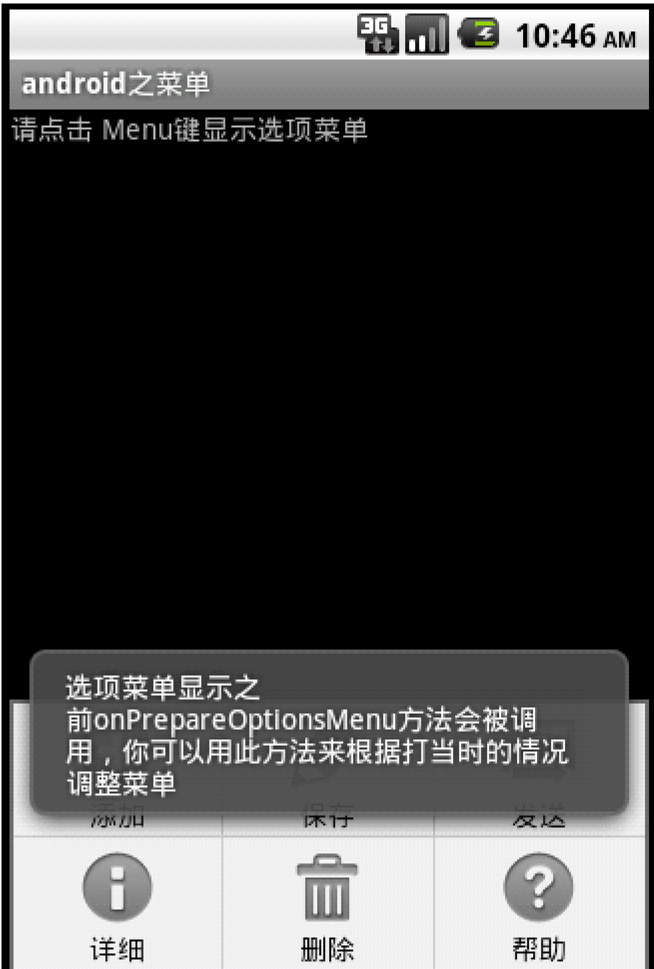


图 3-23　执行效果

3.13　实现定时器效果

实例 054	使用 Chronometer 在屏幕中实现定时器效果
源码路径	光盘:\daima\054
视频路径	光盘:\视频\054
实例必备	054.显式 Intent 和隐式 Intent.pdf ① 显式 Intent（Explicit Intent）的基本用法 ② 隐式 Intent（Implicit Intent）的基本用法

3.13.1　实例说明

定时器确实是一项了不起的发明，使相当多需要人工控制时间的工作变得简单了许多。人们甚至

将定时器用在了军事方面，制成了定时炸弹、定时雷管。现在很多家用电器都安装了定时器来控制开关或工作时间。在本实例中，将使用 Chronometer 控件在 Android 屏幕中实现定时器效果。

Chronometer 是一个简单的定时器，可以指定一个开始时间，并以此定时，或者如果不指定一个开始时间，它将会使用通话开始时的时间。默认情况下会显示在当前定时器的值的形式为“分:秒”或“hh:mm:ss”，或者可以使用的 Set（字符串）格式的定时器值到一个任意字符串。

1. 属性

android:format：此属性用于定义时间的格式，如 hh:mm:ss。

2. 方法

- ☑ setBase(long base)：设置倒计时定时器。
- ☑ setFormat(String format)：设置显示时间的格式。
- ☑ start()：开始计时。
- ☑ stop()：停止计时。
- ☑ setOnChronometerTickListener(Chronometer.OnChronometerTickListener listener)：当计时器改变时调用。

3.13.2 具体实现

（1）编写 XML 布局文件，在其中插入 5 个 Button 控件，通过按钮可以控制计时控件。

（2）编写程序文件 ChronometerDemo.java，分别设置单击 Button 按钮后的处理事件。主要代码如下所示。

```
public class ChronometerDemo extends Activity {
private Chronometer mChronometer;
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.chronometerpage);
  Button button;
        mChronometer = (Chronometer) findViewById(R.id.chronometer);
        button = (Button) findViewById(R.id.start);
        button.setOnClickListener(mStartListener);
        button = (Button) findViewById(R.id.stop);
        button.setOnClickListener(mStopListener);
        button = (Button) findViewById(R.id.reset);
        button.setOnClickListener(mResetListener);
        button = (Button) findViewById(R.id.set_format);
        button.setOnClickListener(mSetFormatListener);
        button = (Button) findViewById(R.id.clear_format);
        button.setOnClickListener(mClearFormatListener);
    }
    View.OnClickListener mStartListener = new OnClickListener() {
        public void onClick(View v) {
            mChronometer.start();
        }
    };
```

```
View.OnClickListener mStopListener = new OnClickListener() {
    public void onClick(View v) {
        mChronometer.stop();
    }
};
View.OnClickListener mResetListener = new OnClickListener() {
    public void onClick(View v) {
        mChronometer.setBase(SystemClock.elapsedRealtime());
    }
};
View.OnClickListener mSetFormatListener = new OnClickListener() {
    public void onClick(View v) {
        mChronometer.setFormat("Formatted time (%s)");
    }
};
View.OnClickListener mClearFormatListener = new OnClickListener() {
    public void onClick(View v) {
        mChronometer.setFormat(null);
    }
};
```

执行后的效果如图 3-24 所示。



图 3-24　执行效果

第 4 章　界面显示实战

消费者在购买手机时，往往既要求界面美观，又要求功能强大。所以对于 Android 程序员来说，工作任务就是开发界面美观绚丽且功能强大的应用程序。本章将以第 1 章和第 2 章的内容为基础，详细讲解在界面中显示屏幕元素的基本知识。

4.1　获取屏幕的分辨率

实例 055	获取手机屏幕的分辨率
源码路径	光盘:\daima\055
视频路径	光盘:\视频\055
实例必备	055.IntentFilter 详解.pdf ① IntentFilter 基础 ② IntentFilter 响应隐式 Intent ③ Android 解析 IntentFilter

4.1.1　实例说明

不同的手机有不同的分辨率，分辨率越高，屏幕越清晰。在当前手机市场中，屏幕的分辨率已经成为划分手机档次的一个标准。在开发手机应用程序时，除了底层对 API 的掌握度之外，最重要的仍是对屏幕分辨率的概念，因各家手机厂商所采用的屏幕尺寸不同，用户 UI 接口呈现及布局自然也各异。

尽管 Android 可设置随着窗口大小调整缩放比例，但即便如此，手机程序设计人员还是必须知道手机屏幕的边界，以避免缩放造成的布局（Layout）变形问题。这个实例非常简单，只需几行程序即可获取手机的分辨率，其关键是 DisplayMetrics 类的应用。

4.1.2　具体实现

编写主程序文件，因为在 android.util 下的 DisplayMetrics 对象中记录了一些常用的信息，如显示信息、屏幕大小、维度和字体等。所以在此文件中可以使用 DisplayMetrics 对象获取当前手机屏幕的分辨率，在使用时要引用 android.util.DisplayMetrics。其中，DisplayMetrics 对象中的 widthPixels 和 heightPixels 字段是整数类型。主程序文件的主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
```

```
    /*必须引用 android.util.DisplayMetrics*/
    DisplayMetrics dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);

    String strOpt = "分辨率是：" +
            dm.widthPixels + " × " + dm.heightPixels;

    mTextView01 = (TextView) findViewById(R.id.myTextView01);
    mTextView01.setText(strOpt);
}
```

运行后的效果如图 4-1 所示。

上述程序一开始所创建的 DisplayMetrics 对象（程序中的 dm）不需要传递任何参数（构造时），但在调用 getWindowManager()之后，会取得现有的 Activity 的 Handler，此时，调用 getDefaultDisplay()方法将取得的宽高维度存放于 DisplayMetrics 对象 dm 中，而取得的宽高维度是以像素为单位（Pixel），"像素"所指的是"绝对像素"而不是"相对像素"。

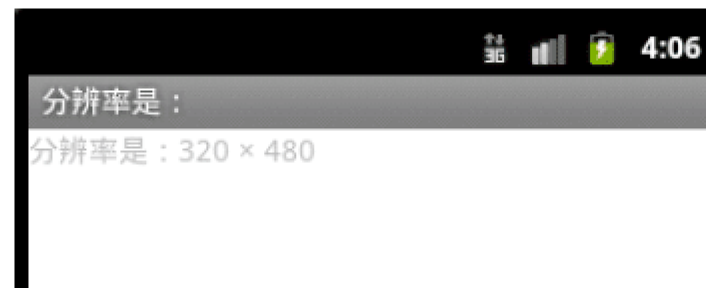


图 4-1　运行效果

4.2　设置显示文字的样式

实例 056	设置屏幕中的文字样式
源码路径	光盘:\daima\056
视频路径	光盘:\视频\056
实例必备	056.Component 属性.pdf

4.2.1　实例说明

前几个实例讲解了某个单独对象的修饰、处理方法，但逐个指定文字的大小、颜色很浪费时间，因此可以使用类似 CSS 样式的方法来指定颜色、大小。本节将通过一个简单实例的实现过程，介绍样式化处理对象的方法。

4.2.2　具体实现

（1）编写主程序文件，此文件代码十分简单，只是加载了 R.layout.main 定义的布局内容而已，但由于定义在 main.xml 中的语句不同，自然也有不同的显示效果，主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
}
```

（2）编写布局文件，本实例的布局文件由 main.xml 和 style.xml 共同实现。其中，文件 main.xml 是纯布局文件，设置了初始化 TextView 时使用指定的 Style 属性。文件 main.xml 的实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:background="@drawable/white"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <!-- 应用模式 1 的 TextView -->
  <TextView
  style="@style/DavidStyleText1"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:gravity="center_vertical|center_horizontal"
  android:text="@string/str_text_view1"
  />
  <!-- 应用模式 2 的 TextView -->
  <TextView
  style="@style/DavidStyleText2"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:gravity="center_vertical|center_horizontal"
  android:text="@string/str_text_view2"
  />
</LinearLayout>
```

（3）编写文件 style.xml，在其中定义文本的显示样式。主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="DavidStyleText1">
    <item name="android:textSize">48sp</item>
    <item name="android:textColor">#EC9237</item>
  </style>
  <style name="DavidStyleText2">
    <item name="android:textSize">24sp</item>
    <item name="android:textColor">#FF7F7C</item>
    <item name="android:fromAlpha">0.0</item>
    <item name="android:toAlpha">0.0</item>
  </style>
</resources>
```

运行后的效果如图 4-2 所示。

对Text1起作用

对Text2起作用

图 4-2　运行效果

在本实例中创建了两个 TextView 以做对比，这样就呈现出两种不同的样式，而 Style 的写法与先前介绍到的颜色常数（color.xml）相同，同样是定义在 res\values 目录下，但其 XML 定义的方式不同。

4.3　实现屏幕界面的转换

实例 057	实现屏幕界面的转换
源码路径	光盘:\daima\057
视频路径	光盘:\视频\057
实例必备	057.Action 属性.pdf

4.3.1　实例说明

在网络冲浪过程中会经常浏览不同的页面，在使用手机时也经常访问不同的屏幕界面。本实例将详细介绍实现屏幕界面转换处理的方法。

使用计算机浏览网页时，想要在两个网页间转换，只要利用超链接（HyperLink）即可实现。那么在手机中如何实现页面之间的转换呢？最简单的方式是改变 Activity 的 Layout。在本实例中布局两个 Layout，分别是 Layout1（main.xml）和 Layout2（mylayout.xml），默认载入的 Layout 是 main.xml，在 Layout1 中创建一个按钮，当单击该按钮后会显示第二个 Layout（mylayout.xml）界面。同样在 Layout2 界面中也设置了一个按钮，当单击该按钮后会返回到原来的 Layout1 界面。

4.3.2　具体实现

（1）编写主程序文件，在此文件中设置预加载的 Layout 布局是 main.xml，屏幕上显示的是黑色背景的“演示界面转换”，当单击第一个 Layout 中的按钮时会改变 Activity 的 Layout 为 layout2.xml，并且屏幕上显示变为白色背景文字“083”。主程序文件的主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  /*载入 main.xml Layout*/
  setContentView(R.layout.main);

  /*以 findViewById()取得 Button 对象，并添加 onClickListener 事件*/
  Button b1 = (Button) findViewById(R.id.button1);
  b1.setOnClickListener(new Button.OnClickListener()
  {
    public void onClick(View v)
    {
      jumpToLayout2();
    }
  });
}
/*将 layout 由 main.xml 切换成 layout2.xml*/
public void jumpToLayout2()
{
  /*将 layout 改成 mylayout.xml*/
  setContentView(R.layout.layout2);
```

```
        /*通过 findViewById()取得 Button 对象，并添加 onClickListener 事件*/
        Button b2 = (Button) findViewById(R.id.button2);
        b2.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                jumpToLayout1();
            }
        });
    }

    /*将 layout 由 mylayout.xml 切换成 main.xml*/
    public void jumpToLayout1()
    {
        /*将 layout 改成 main.xml*/
        setContentView(R.layout.main);

        /*以 findViewById()取得 Button 对象，并添加 onClickListener 事件*/
        Button b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(new Button.OnClickListener()
        {
            public void onClick(View v)
            {
                jumpToLayout2();
            }
        });
    }
}
```

（2）编写修饰文件，本实例的布局文件由 main.xml 和 mylayout.xml 共同实现。其中，文件 main.xml 是为了突出显示 Layout 间切换的效果，改变两个 Layout 的背景色及输出文字。在 main.xml 中定义其背景色为黑色，输出文字为“演示界面转换”；而在 mylayout.xml 中，定义了其背景色为白色，输出文字为“083”。

运行后的效果如图 4-3 所示。当单击屏幕中的“去 Layout2”按钮后，屏幕上的文本将会发生变化，由原来的“演示界面转换”变为“083”，如图 4-4 所示。



图 4-3　运行效果



图 4-4　转换后效果

4.4　在一个 Activity 中调用另一个 Activity

实例 058	在一个 Activity 中调用另一个 Activity
源码路径	光盘:\daima\058
视频路径	光盘:\视频\058
实例必备	058.Category 属性.pdf

4.4.1　实例说明

在 Android 开发应用中，通过切换 Layout 的方式可以实现在手机屏幕中切换界面。除了可以在页面中转换背景、颜色或文字内容外，还可以实现对 Activity 界面的置换。这不是仅改变 Layout 就能完成的，因为在置换时需要传递的变量不像在网页中那样通过 Cookie 或 Session 实现，需要在程序中将主控权转交到另外一个 Activity，只靠前面介绍的 Layout 技术是无法实现的。通过本实例的实现过程，详细介绍一个 Activity 调用另一个 Activity 的方法。

4.4.2　具体实现

（1）编写主程序文件 zhihuan1.java，在此文件按钮中调用另一个 Activity（zhihuan_1），调用 finish()将 Activity 关闭，接着将主控权交给 Activity2。主程序文件 zhihuan1.java 的主要实现代码如下所示。

```
public class zhihuan1 extends Activity
{
  /** Called when the activity is first created.*/
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /*载入 mylayout.xml Layout*/
    setContentView(R.layout.main);

    /*以 findViewById()取得 Button 对象，并添加 onClickListener*/
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /*new 一个 Intent 对象，并指定要启动的 class*/
        Intent intent = new Intent();
        intent.setClass(zhihuan1.this, zhihuan1_1.class);

        /*调用一个新的 Activity*/
        startActivity(intent);
        /*关闭原本的 Activity*/
        zhihuan1.this.finish();
      }
    });
  }
}
```

（2）编写文件 zhihuan1_1.java，此文件是第二个 Activity 的主程序，其加载的 Layout 为 mylayout.xml，屏幕上所显示的是白色背景的“这是第二个 Activity”，在此 Activity（Activity2）界面单击按钮后会重新调用 Activity1，并使用 finish()方法关闭 Activity2，主要实现代码如下所示。

```
public class zhihuan1_1 extends Activity
{
  /** Called when the activity is first created.*/
```

```
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      /*载入 main.xml Layout*/
      setContentView(R.layout.layout2);

      /*以 findViewById()取得 Button 对象，并添加 onClickListener*/
      Button b2 = (Button) findViewById(R.id.button2);
      b2.setOnClickListener(new Button.OnClickListener()
      {
        public void onClick(View v)
        {
          /*new 一个 Intent 对象，并指定要启动的 class*/
          Intent intent = new Intent();
          intent.setClass(zhihuan1_1.this, zhihuan1.class);

          /*调用一个新的 Activity*/
          startActivity(intent);
          /*关闭原本的 Activity*/
          zhihuan1_1.this.finish();
        }
      });
    }
}
```

（3）编写修饰文件，本实例的布局文件是 main.xml 和 mylayout.xml。其中，文件 main.xml 是为了突出显示 Layout 间切换的效果，特意有区别地输出两个 Layout 的背景和文字。

（4）编写配置文件，因为在该实例中添加了一个 Activity，所以必须在文件 AndroidManifest.xml 中定义一个新的 Activity，并命名为 name，否则程序将无法编译运行，其主要实现代码如下所示。

```
    <application android:icon="@drawable/icon" android:label="@string/app_name">
      <activity android:name=".zhihuan1"
                    android:label="@string/app_name">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
      <activity android:name="zhihuan1_1"></activity>
```

运行后的效果如图 4-5 所示。当单击屏幕中的按钮后，屏幕上的文本将会发生变化，如图 4-6 所示。



图 4-5 运行效果



图 4-6 转换后效果

系统中新添加 Activity 时，必须在 AndroidManifest.xml 中定义一个新的 Activity，否则系统将会因

为找不到 Activity 而发生编译错误。定义代码如下所示。

```
<activity android:name="zhihuan_1"></activity>
```

4.5　改变显示文字的颜色

实例 059	单击按钮后改变文字颜色
源码路径	光盘:\daima\059
视频路径	光盘:\视频\059
实例必备	059.Data 属性和 Type 属性.pdf

4.5.1　实例说明

经过本书前面实例的学习，了解了多种和 TextView 文字相关的处理方法，也了解了和按钮相关的处理方法。本实例将对前面的知识进行整合处理，通过 setOnClickListener 监听单击事件，在单击后触发函数 setTextColor 来改变文本颜色。因为提前创建一个字形定义的颜色数组 mColor，在单击按钮时会根据此颜色数组索引值的变化来置换 TextView 中的文字颜色。

4.5.2　具体实现

编写主程序文件，在此文件中使用函数 setTextColor 来改变文字的颜色，单击按钮后执行函数 onClick，此函数驱动了 setTextColor 的事件。主程序文件的主要实现代码如下所示。

```
private Button mButton;
private TextView mText;
private int[ ] mColors;
private int colornum;
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  /*通过 findViewById 构造器来使用 main.xml 与 string.xml 中的 button 和 textView 的参数*/
  mButton=(Button) findViewById(R.id.mybutton);
  mText= (TextView) findViewById(R.id.mytext);
  /*声明并构造一整数 array 来存储欲使用的文字颜色*/
  mColors = new int[ ]  {
  Color.BLACK, Color.RED, Color.BLUE,
  Color.GREEN, Color.MAGENTA, Color.YELLOW
                      };
  colornum=0;
  /*使用 setOnClickListener 让按钮聆听事件*/
  mButton.setOnClickListener(new View.OnClickListener()   {
    /*使用 onClick 让用户单击按钮来驱动改变文字颜色*/
    public void onClick(View v)    {
```

```
            if (colornum < mColors.length) {
               mText.setTextColor(mColors[colornum]);
               colornum++;
            }
            else
               colornum=0;
         }
      });
   }
}
```

运行后的效果如图 4-7 所示；当单击“单击”按钮后会改变文本的颜色，如图 4-8 所示；再次单击“单击”按钮后，文本还会改变颜色，如图 4-9 所示。



图 4-7　运行效果



图 4-8　改变颜色



图 4-9　再次改变颜色

4.6　在屏幕中实现拖动图片特效

实例 060	在手机屏幕中实现拖动图片特效
源码路径	光盘:\daima\060
视频路径	光盘:\视频\060
实例必备	060.Extra 属性.pdf

4.6.1　实例说明

很多手机产品可以在屏幕中通过滑动的方式拖动一幅图片，这样的效果很吸引用户的眼球。在 Android 中可以通过类 Android.content.Context、Android.widget.BaseAdapter 和 Android.widget.ImageView 来实现拖动图片特效。在 Activity 中，Context 作为 Android.content 的子类，好比 Canvas 画布随时会被处理或覆盖。在本实例的 Layout 中布局一个 Gallery 对象，然后通过类 widget.BaseAdapter 作为容器存放 Gallery 所需要的图。为了复习前面介绍的 Gallery 的使用方法，在实例中使用了 Android 的 Icon 图标。

4.6.2　具体实现

编写主程序文件，在此创建一个继承于 BaseAdapte 的方法 ImageAdapter()，此方法的功能是暂时保存要显示的图片作为 Gallery 控件图片的引用。主程序文件的主要代码如下所示。

```
public class texiao089 extends Activity{
   private TextView mTextView01;
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mTextView01 = (TextView) findViewById(R.id.myTextView01);
    mTextView01.setText(getString(R.string.str_txt1));
    mTextView01.setTextColor(Color.BLUE);

    ((Gallery) findViewById(R.id.myGallery1))
                    .setAdapter(new ImageAdapter(this));
  }
  public class ImageAdapter extends BaseAdapter
  {
    /*类成员 myContext 为 Context 父类*/
    private Context myContext;

    /*使用 android.R.drawable 中的图片作为图库来源，类型为整数数组*/
    private int[ ] myImageIds =
                        {
                          android.R.drawable.btn_minus,
                          android.R.drawable.btn_radio,
                          android.R.drawable.ic_lock_idle_low_battery,
                          android.R.drawable.ic_menu_camera
                        };
    /*构造器只有一个参数，即要存储的 Context*/
    public ImageAdapter(Context c) { this.myContext = c; }
    /*返回所有已定义的图片总数量*/
    public int getCount() { return this.myImageIds.length; }
    /*利用 getItem()方法，获取目前容器中图像的数组 ID*/
    public Object getItem(int position) { return position; }
    public long getItemId(int position) { return position; }

    /*取得目前欲显示的图像 View，传入数组 ID 值使之读取与成像*/
    public View getView(int position, View convertView,
                              ViewGroup parent)
    {
      /*创建一个 ImageView 对象*/
      ImageView i = new ImageView(this.myContext);

      i.setImageResource(this.myImageIds[position]);
      i.setScaleType(ImageView.ScaleType.FIT_XY);

      /*设置该 ImageView 对象的宽高，单位为 dip*/
      i.setLayoutParams(new Gallery.LayoutParams(120, 120));
      return i;
    }
    /*依据距离中心的位移量，利用 getScale 返回 views 的大小(0.0f to 1.0f)*/
    public float getScale(boolean focused, int offset)
    {
      /*Formula: 1 / (2 ^ offset)*/
      return Math.max(0,1.0f/(float)Math.pow(2,Math.abs(offset)));
    }
  }
}
```

运行后会在屏幕中显示几幅图片，当单击某个图片后会以滚动特效的样式显示，如图 4-10 所示。



图 4-10　运行效果

4.7　在屏幕中实现一个 About（关于）信息效果

实例 061	在屏幕中实现一个 About（关于）信息效果
源码路径	光盘:\daima\061
视频路径	光盘:\视频\061
实例必备	061.Flag 属性.pdf

4.7.1　实例说明

在计算机系统中经常遇到 About（关于）信息，主要功能是说明当前软件或硬件的基本信息，如网站中的“关于我们”。在本实例中，通过手机接口 Menu Shotcut 实现了“关于”对话框和“离开”对话框效果。在程序中除了默认覆盖的 onCreate 响应函数外，还建立了两个类函数 onCreateMenu 和 onOptionItemSelected。前者用于创建 Menu 菜单项目，后者则用于处理菜单被选择运行后的事件。

4.7.2　具体实现

编写主程序文件，在此文件中创建了一个类函数 onCreateOptionsMenu(Menu menu)，此函数用于添加 Menu 菜单，然后使用函数 onOptionItemSelected 获取菜单选择项目处理对应的事件，用 getItemId()=0 表示“关于”，用 getItemId()=1 表示“离开”。主程序文件的主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
}

public boolean onCreateOptionsMenu(Menu menu)
{
  menu.add(0, 0, 0, R.string.app_about);
  menu.add(0, 1, 1, R.string.str_exit);
  return super.onCreateOptionsMenu(menu);
}
public boolean onOptionsItemSelected(MenuItem item)  {
  super.onOptionsItemSelected(item);
```

```
        switch(item.getItemId())
        {
          case 0:
            openOptionsDialog();
            break;
          case 1:
            finish();
            break;
        }
        return true;
    }
    private void openOptionsDialog() {
        new AlertDialog.Builder(this)
        .setTitle(R.string.app_about)
        .setMessage(R.string.app_about_msg)
        .setPositiveButton(R.string.str_ok,
            new DialogInterface.OnClickListener(){
             public void onClick(DialogInterface dialoginterface, int i){
             }
            }
            )
        .show();
    }
}
```

运行后的效果如图 4-11 所示，当单击 Menu 按钮后会弹出两个子菜单，如图 4-12 所示；当单击“关于我们”按钮后会弹出一个对话框，此对话框就是 About（关于）的说明信息，如图 4-13 所示。



图 4-11　运行结果

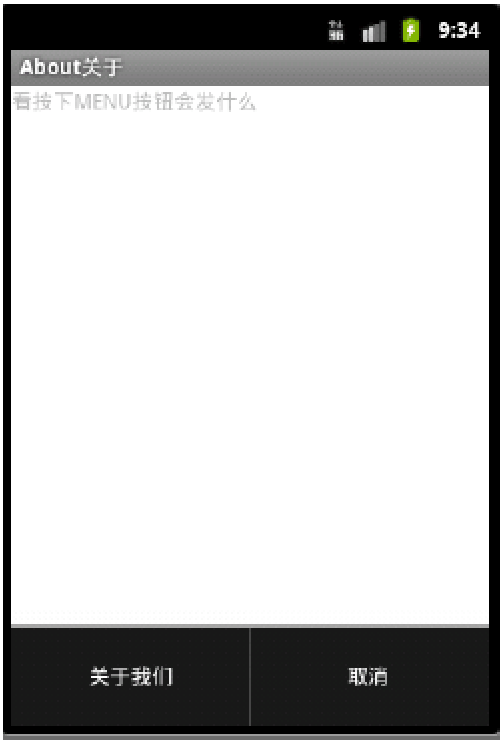


图 4-12　单击 Menu 按钮后



图 4-13　“关于我们”对话框

4.8　实现程序加载效果

实例 062	在手机屏幕中实现程序加载效果
源码路径	光盘:\daima\062
视频路径	光盘:\视频\062
实例必备	062.Intent 和 Activity.pdf

4.8.1 实例说明

在很多应用软件程序中都有程序加载功能，例如系统提示“程序正在加载，请用户慢慢等待……”。在本实例中设计了一个按钮，单击按钮后开始线程的周期，在运行过程中显示 ProgressDialog，线程运行完毕后，结束 ProgressDialog 对话框。

4.8.2 具体实现

编写主程序文件，在此文件中创建了一个按钮对象，单击按钮后会触发 myShowProgressBar 事件以更改 TextView 中的文字，并将焦点传递给前台的 ProgressDialog.show。运行 3 秒钟后，再将焦点传递返回给原来的 Activity。主程序文件的主要代码如下所示。

```
private Button mButton1;
private TextView mTextView1;
public ProgressDialog myDialog = null;
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mButton1 =(Button) findViewById(R.id.myButton1);
  mTextView1 = (TextView) findViewById(R.id.myTextView1);
  mButton1.setOnClickListener(myShowProgressBar);
}
Button.OnClickListener myShowProgressBar =
new Button.OnClickListener() {
  public void onClick(View arg0)
  {
    final CharSequence strDialogTitle =
          getString(R.string.str_dialog_title);
    final CharSequence strDialogBody =
          getString(R.string.str_dialog_body);
    //显示 Progress 对话框
    myDialog = ProgressDialog.show(
                 dengdai092.this,
                 strDialogTitle,
                 strDialogBody,
                 true
               );
    mTextView1.setText(strDialogBody);
    new Thread(){
      public void run()  {
        try{
          /*暂停 3 秒作为示范*/
          sleep(3000);
        }
        catch (Exception e)
        {
```

```
                e.printStackTrace();
            }
            finally{
                //卸载所创建的 myDialog 对象
                myDialog.dismiss();
            }
        }
    }.start(); /*开始运行线程*/
  }
 };
}
```

运行后的效果如图 4-14 所示，单击“按下后开始执行前台运算”按钮后弹出“等待”提示框信息，如图 4-15 所示。

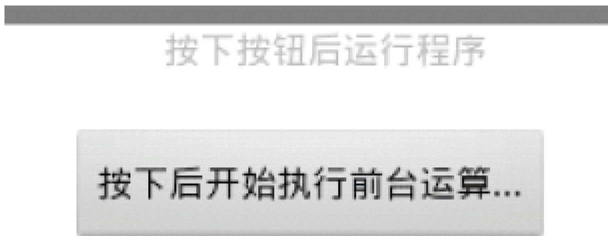


图 4-14　运行效果

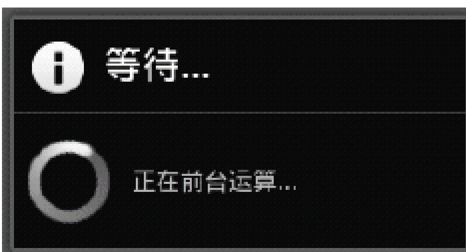


图 4-15　显示等待提示信息

4.9　实现一个有选择项的对话框

实例 063	创建一个有选择项的对话框
源码路径	光盘:\daima\063
视频路径	光盘:\视频\063
实例必备	063.显式启动新的 Activity.pdf

4.9.1　实例说明

对话框是网络中常见的应用，手机软件中也必不可少。在本书前面的实例中已经多次介绍了生成对话框的方法，也介绍了 AlertDialog.Builder 对话框的用法。其实在 AlertDialog.Builder 对话框中还可以包含多个子对话框。在本实例中创建了一个按钮事件，触发该按钮事件后以类似列表项目的方式呈现在 AlertDialog 中。

4.9.2　具体实现

编写主程序文件，自此文件中设置了一个 Button 对象，单击后会弹出一个拥有 3 个选项的对话框。主程序文件的主要代码如下所示。

```
Button.OnClickListener myShowAlertDialog = new Button.OnClickListener(){
  public void onClick(View arg0)  {
    new AlertDialog.Builder(xuanze093.this)
      .setTitle(R.string.str_alert_title)
      .setItems(R.array.items_irdc_dialog,
```

```
            new DialogInterface.OnClickListener()   {
              public void onClick(DialogInterface dialog, int whichcountry){
                CharSequence strDialogBody = getString(R.string.str_alert_body);
                String[ ] aryShop =
                getResources().getStringArray(R.array.items_irdc_dialog);
                  new AlertDialog.Builder(xuanze093.this)
                  .setMessage(strDialogBody + aryShop[whichcountry])
                  .setNeutralButton(R.string.str_ok, new DialogInterface.OnClickListener()
                  {
                    public void onClick(DialogInterface dialog, int whichButton)
                    {

                    }
                  })
                  .show();
              }
            })
            .setNegativeButton("鍙栨", new DialogInterface.OnClickListener()
            {
              @Override
              public void onClick(DialogInterface d, int which)
              {
                d.dismiss();
              }
            })
            .show();
        } /*End: public void onClick(View arg0)*/
    };
}
```

运行后的效果如图 4-16 所示，单击“单击我”按钮后弹出一个选择项菜单，如图 4-17 所示；选择一个选项后会弹出一个对话框，如图 4-18 所示。单击 OK 按钮后返回到图 4-16 所示的初始界面。



图 4-16　运行效果



图 4-17　选择项



图 4-18　选择提示

4.10　改变手机的主题

实例 064	改变手机的主题
源码路径	光盘:\daima\064
视频路径	光盘:\视频\064
实例必备	064.隐式 Intent 和运行时绑定.pdf

4.10.1　实例说明

改变显示主题是常见的手机应用程序之一，在大多数手机中用户可以选择自己需要的主题风格。本书前面的内容中已经介绍过使用 Style 的方法，通过该方法可以开发出不同外观的显示效果，并且还可以方便对程序的维护。引入 Style 后，大大改善了程序员和视觉设计人员之间存在的沟通问题。在 Android 开发应用中，除了可以使用 Style 设置不同主题外，还可以针对每个 Activity、前景、背景、透明度等进行设置规划。

4.10.2　具体实现

（1）编写主程序文件，在此文件中使用方法 setTheme()指定了 Activity 界面的主题，其中，主题设置文件在 Style.xml 中。主程序文件的主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   /*
   * 应用透明背景的主题
   * setTheme(R.style.Theme_Transparent);
   */
   /*
     * 应用布景主题 1
   */
      setTheme(R.style.Theme_Translucent);
   /*
     * 应用布景主题 2
   * setTheme(R.style.Theme_Translucent2);
   */
   setContentView(R.layout.main);
}
```

（2）编写主题文件 Style.xml，在里面预先设置了 Theme、ThemeTranslucent、ThemeTransparent 和 TextAppearance.Theme.PlaneText 4 种主题样式。

运行后的效果如图 4-19 所示。



图 4-19　运行效果

4.11 自动显示输入的数据

实例 065	在屏幕中自动显示输入的数据
源码路径	光盘:\daima\065
视频路径	光盘:\视频\065
实例必备	065.Activity 的返回值.pdf ① 启动子 Activity ② 返回值 ③ 处理子 Activity 的结果

4.11.1 实例说明

本实例提供了一个文本输入框，如果用户输入一个电话号码，则可以进行拨号处理；如果输入一个网址，则可以登录到这个地址；如果输入一个邮箱地址，则可以发送邮件。

上述功能是通过对象 Linkify 实现的，通过此对象可以让系统动态获取输入的数据，对获取的数据作出判断，判断其是电话、邮箱还是网址，并作出对应的处理。只需通过 TextView 和 EditText 交互，就可以在屏幕中显示输入的数据。在具体实现上，只需重写 EditText.setOnkeyListener()即可。

4.11.2 具体实现

编写主程序文件，在此文件中设置了 Linkify 的处理事件，必须在创建 TextView 之后设置 Linkify，否则设置的回复处理不会起作用，还设置 mTextView01 拥有自动连接功能，这样就能自动跳转到指定网址的页面。主程序文件的主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
  {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  mTextView01 = (TextView)findViewById(R.id.myTextView1);
  mEditText01 = (EditText)findViewById(R.id.myEditText1);
  mEditText01.setOnKeyListener(new EditText.OnKeyListener()
  {
    @Override
    public boolean onKey(View arg0, int arg1, KeyEvent arg2)
    {
     mTextView01.setText(mEditText01.getText());
      /*判断输入的是何种类型，并与系统连接*/
     Linkify.addLinks(mTextView01,Linkify.WEB_URLS|Linkify.
          EMAIL_ADDRESSES|Linkify.PHONE_NUMBERS);
      return false;
    }
  });
  }
```

执行后的效果如图 4-20 所示，在文本框中可以输入电话号码、邮箱地址或网址，输入后可显示对应的操作。例如，输入 www.163.com 后，会在下面自动显示输入的网址，如图 4-21 所示；当单击网址后会打开对应的页面，如图 4-22 所示。

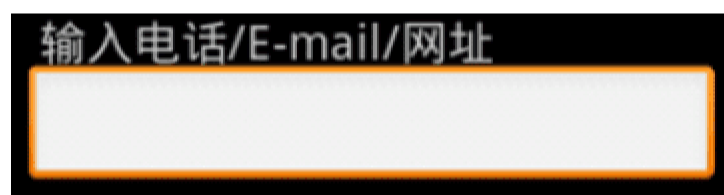


图 4-20　执行效果



图 4-21　自动显示输入的文本



图 4-22　来到网易主页

4.12　实现图文提醒功能

实例 066	实现图文提醒功能
源码路径	光盘:\daima\066
视频路径	光盘:\视频\066
实例必备	066.Android 本地动作.pdf

4.12.1　实例说明

在本书前面的实例中，已经讲解过 Toast 实现提醒功能的基本知识，通过 Toast 能够在手机中实现一个醒目的提示效果。在本实例的 Toast 中放置一幅图片和一行文字，实现图文并茂的提醒效果。图文提醒和单独的文字提醒相比，具有更好的视觉冲击效果。在具体实现上，在 Toast 中放置一个 Layout，其中包含了图片和文字，这些图片和文字就是提醒的内容。

4.12.2　具体实现

本实例主程序文件的主要实现代码如下所示。

```
private Button mButton01;
@Override
public void onCreate(Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mButton01 = (Button)findViewById(R.id.myButton1);
        /*设置 Button 用 OnClickListener 启动事件*/
        mButton01.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                // TODO Auto-generated method stub
                /*创建 ImageView*/
                ImageView mView01 = new ImageView(example097.this);
                TextView mTextView = new TextView(example097.this);
                /*创建 LinearLayout 对象*/
                LinearLayout lay = new LinearLayout(example097.this);

                /*设置 mTextView 以获取 string 值*/
                mTextView.setText(R.string.app_url);
                /*判断 mTextView 的内容，并与系统连接*/
                Linkify.addLinks
                (
                    mTextView,Linkify.WEB_URLS|
                    Linkify.EMAIL_ADDRESSES|
                    Linkify.PHONE_NUMBERS
                );
                /*用 Toast 提示*/
                Toast toast = Toast.makeText
                              (
                                  example097.this,
                                  mTextView.getText(),
                                  Toast.LENGTH_LONG
                              );
                /*自定义 View 对象*/
                View textView = toast.getView();
                /*以水平方式排列*/
                lay.setOrientation(LinearLayout.HORIZONTAL);
                /*在 ImageView Widget 中指定显示的图片*/
                mView01.setImageResource(R.drawable.icon);
                /*在 Layout 中添加刚创建的 View*/
                lay.addView(mView01);
                /*在 Toast 中显示文字*/
                lay.addView(textView);

                /*以 Toast 和 setView 方法传入 Layout*/
                toast.setView(lay);
                /*显示 Toast 提示*/
                toast.show();
            }
        });
    }
}
```

上述代码的实现流程如下所示。

（1）设置用 OnClickListener 监听 Button 按钮，当按钮被单击时启动执行事件 setOnClickListener。

（2）分别创建 ImageView 对象和 LinearLayout 对象，并设置用 mTextView 抓取 string 值。

（3）判断 mTextView 的内容，并与系统实现连接。

（4）用 Toast 提醒的方式将内容显示出来。

执行后的效果如图 4-23 所示，单击“图文提醒”按钮后弹出一个图文效果的提醒，如图 4-24 所示。



图 4-23　初始效果



图 4-24　图文提醒

4.13　实现 QQ 状态栏效果

实例 067	实现 QQ 状态栏效果
源码路径	光盘:\daima\067
视频路径	光盘:\视频\067
实例必备	067.广播事件.pdf

4.13.1　实例说明

手机的最顶端是状态栏，通常显示时间、信号强度和电池电量，用户可以在状态栏中显示一幅图片，并结合图片和文字来实现更加美观的提示。在本实例的状态栏中放置一幅图标作为提醒，在具体实现时，使用 Android API 管理提示信息，并且定义了通知管理对象 NotificationManage，将 Notification 添加到 NotificationManage 以便将信息显示在状态栏。

4.13.2　具体实现

（1）编写主程序文件，此文件的具体实现流程如下所示。

① 分别设置“在线中”“离开了”“忙碌中”“马上回来”“离线中” 5 种状态，具体代码如下所示。

```
/*声明对象变量*/
private NotificationManager myNotiManager;
private Spinner mySpinner;
private ArrayAdapter<String> myAdapter;
```

```
private static final String[ ] status =
{ "在线中","离开了","忙碌中","马上回来","离线中" };

@Override
protected void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  /*载入 main.xml Layout*/
  setContentView(R.layout.main);
```

② 初始化对象，然后使用 myspinner_dropdown 自定义下拉菜单模式，并将 ArrayAdapter 添加到 Spinner 对象中，根据具体状态显示对应的提示图标，还定义了 onNothingSelected(Adapter View<?> arg0) 供用户选择当前的状态，具体代码如下所示。

```
  /*初始化对象*/
  myNotiManager=
    (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
  mySpinner=(Spinner)findViewById(R.id.mySpinner);
  myAdapter=new ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,status);
  /*应用 myspinner_dropdown 自定义下拉菜单模式*/
  myAdapter.setDropDownViewResource(R.layout.myspinner);
  /*将 ArrayAdapter 添加到 Spinner 对象中*/
  mySpinner.setAdapter(myAdapter);
  /*将 mySpinner 添加到 OnItemSelectedListener*/
  mySpinner.setOnItemSelectedListener(
    new Spinner.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> arg0,View arg1,
                               int arg2,long arg3)
    {
      /*依照选择的 item 来判断要发哪一个 notification*/
      if(status[arg2].equals("在线中"))
      {
        setNotiType(R.drawable.msn,"在线中");
      }
      else if(status[arg2].equals("离开了"))
      {
        setNotiType(R.drawable.away,"离开了");
      }
      else if(status[arg2].equals("忙碌中"))
      {
        setNotiType(R.drawable.busy,"忙碌中");
      }
      else if(status[arg2].equals("马上回来"))
      {
        setNotiType(R.drawable.min,"马上回来");
      }
      else
      {
        setNotiType(R.drawable.offine,"离线中");
      }
    }
```

```
            @Override
            public void onNothingSelected(AdapterView<?> arg0)
            {
            }
        });
    }
```

③ 定义方法 setNotiType()，创建新的 Intent 和 Notication，并设置相关对应的参数，主要代码如下所示。

```
    /*发出 Notification 的方法*/
    private void setNotiType(int iconId, String text)
    {
      /*创建新的 Intent，作为单击 Notification 留言条时运行的 Activity*/
      Intent notifyIntent=new Intent(this,example098_1.class);
      notifyIntent.setFlags( Intent.FLAG_ACTIVITY_NEW_TASK);
      /*创建 PendingIntent 作为设置递延运行的 Activity*/
      PendingIntent appIntent=PendingIntent.getActivity(example098.this,
                                    0,notifyIntent,0);
      /*创建 Notication 提醒，并设置相关参数*/
      Notification myNoti=new Notification();
      /*设置 statusbar 显示的 icon*/
      myNoti.icon=iconId;
      /*设置 statusbar 显示的文字信息*/
      myNoti.tickerText=text;
      /*设置 notification 发生时同时发出默认声音*/
      myNoti.defaults=Notification.DEFAULT_SOUND;
      /*设置 Notification 留言条的参数*/
      myNoti.setLatestEventInfo(example.this,"登录状态",text,appIntent);
      /*送出 Notification*/
      myNotiManager.notify(0,myNoti);
    }
  }
```

（2）编写文件 example_1.java，引入 example098_1 extends Activity 并发出 Toast 提醒，实现模拟 QQ/MSN 的登录效果，主要代码如下所示。

```
    /*当用户单击 Notification 留言条时，会运行的 Activity*/
    protected void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);

      /*发出 Toast 提醒*/
      Toast.makeText(example_1.this,
                        "模拟实现 MSN、QQ 切换登录状态",
                        Toast.LENGTH_SHORT
                       ).show();
      finish();
    }
  }
```

执行后将首先显示默认的“在线中”状态，并在状态栏中显示在线图标，如图 4-25 所示。在下拉列表框中可以选择其他状态，对应的状态栏也会显示对应的图标，如图 4-26 所示。例如，当单击“离开了”按钮后，会在状态栏中显示对应的提示图标，如图 4-27 所示，这是一个 MSN 的图标。



图 4-25　初始效果



图 4-26　选择其他状态

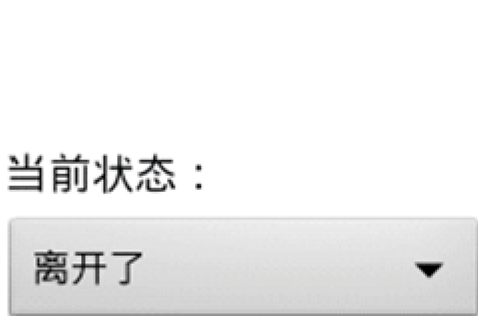


图 4-27　离开图标

4.14　系统文件管理器

实例 068	管理手机系统中的文件
源码路径	光盘:\daima\068
视频路径	光盘:\视频\068
实例必备	068.Broadcast Receiver 监听广播.pdf

4.14.1　实例说明

在 Android 手机应用中，经常利用手机来存储各种各样的文件信息，此时很有必要用一个工具来管理手机系统中的各个文件。在本实例中，通过 ListActivity 和 Java I/O 来查找根目录下的所有文件，并通过 setListAdapter()方法将存放文件信息的 ArrayAdapter 设置给 ListView。

4.14.2　具体实现

（1）编写主程序文件，具体实现流程如下所示。

① 声明变量 items 用于表示存放显示的名称，变量 paths 表示存放文件路径，变量 rootPath 表示起始目录，具体代码如下所示。

```
/*变量声明
   items：存放显示的名称
   paths：存放文件路径
   rootPath：起始目录              */
private List<String> items=null;
private List<String> paths=null;
private String rootPath=" / ";
private TextView mPath;
```

② 载入主布局文件 main.xml，然后初始化 mPath 用于显示目前路径，主要代码如下所示。

```
@Override
protected void onCreate(Bundle icicle)
{
  super.onCreate(icicle);

  /*载入 main.xml Layout*/
  setContentView(R.layout.main);
  /*初始化 mPath，用以显示目前路径*/
```

```
    mPath=(TextView)findViewById(R.id.mPath);
    getFileDir(rootPath);
}
```

③ 定义方法 getFileDir(String filePath)来获取文件的具体架构，具体代码如下所示。

```
/*取得文件架构的方法*/
private void getFileDir(String filePath)
{
  /*设置目前所在路径*/
  mPath.setText(filePath);
  items=new ArrayList<String>();
  paths=new ArrayList<String>();
  File f=new File(filePath);
  File[ ] files=f.listFiles();

  if(!filePath.equals(rootPath))
  {
    /*第一笔设置为“回到根目录”*/
    items.add("b1");
    paths.add(rootPath);
    /*第二笔设置为“回到上一层”*/
    items.add("b2");
    paths.add(f.getParent());
  }
  /*将所有文件添加到 ArrayList 中*/
  for(int i=0;i<files.length;i++)
  {
    File file=files[i];
    items.add(file.getName());
    paths.add(file.getPath());
  }

  /*使用自定义的 MyAdapter 将数据传入 ListActivity*/
  setListAdapter(new MyAdapter(this,items,paths));
}
```

④ 设置单击按钮触发事件处理方法 onListItemClick()，如果是文件夹则运行 getFileDir 函数，如果是文件则运行 openFile 函数，具体代码如下所示。

```
/*设置 ListItem 被单击时要执行的操作*/
@Override
protected void onListItemClick(ListView l,View v,int position,
                                long id)
{
  File file=new File(paths.get(position));
  if (file.isDirectory())
  {
    /*如果是文件夹就运行 getFileDir()*/
    getFileDir(paths.get(position));
  }
  else
  {
    /*如果是文件就运行 openFile()*/
```

```
        openFile(file);
    }
}
```

⑤ 定义方法 openFile(File f)，用于在手机上打开指定的文件，具体代码如下所示。

```
/*在手机上打开文件的方法*/
private void openFile(File f)
{
  Intent intent = new Intent();
  intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
  intent.setAction(android.content.Intent.ACTION_VIEW);

  /*调用 getMIMEType()来获取 MimeType*/
  String type = getMIMEType(f);
  /*设置 intent 的 file 与 MimeType*/
  intent.setDataAndType(Uri.fromFile(f),type);
  startActivity(intent);
}
```

⑥ 定义方法 getMIMEType(File f)来判断文件的类型，其中，end 表示结尾的扩展名，具体代码如下所示。

```
/*判断文件 MimeType 的方法*/
private String getMIMEType(File f)
{
  String type="";
  String fName=f.getName();
  /*取得扩展名*/
  String end=fName.substring(fName.lastIndexOf(".")+1,
                    fName.length()).toLowerCase();

  /*依附文件名的类型决定 MimeType*/
  if(end.equals("m4a")||end.equals("mp3")||end.equals("mid")
      ||end.equals("xmf")||end.equals("ogg")||end.equals("wav"))
  {
    type = "audio";
  }
  else if(end.equals("3gp")||end.equals("mp4"))
  {
    type = "video";
  }
  else if(end.equals("jpg")||end.equals("gif")||end.equals("png")
          ||end.equals("jpeg")||end.equals("bmp"))
  {
    type = "image";
  }
  else
  {
    /*如果无法直接打开，就跳出软件列表供用户选择*/
    type="*";
  }
  type += "/*";
```

```
    return type;
  }
}
```

（2）编写文件 MyAdapter.java，此文件的具体实现流程如下所示。

① 通过如下代码分别声明 4 个变量。

☑　mIcon1：回到根目录的图文件。

☑　mIcon2：回到上一层的图档。

☑　mIcon3：文件夹的图文件。

☑　mIcon4：文件的图档。

```
/*自定义 Adapter，继承于 android.widget.BaseAdapter*/
public class MyAdapter extends BaseAdapter
{
  /*变量声明*/
  private LayoutInflater mInflater;
  private Bitmap mIcon1;
  private Bitmap mIcon2;
  private Bitmap mIcon3;
  private Bitmap mIcon4;
  private List<String> items;
  private List<String> paths;
```

② 定义构造器 MyAdapter，分别传入参数 items、paths 和 mInflater，然后赋值在前面定义的 4 个变量，具体代码如下所示。

```
  /*MyAdapter 的构造器，传入 3 个参数 */
  public MyAdapter(Context context,List<String> it,List<String> pa)
  {
    /*参数初始化*/
    mInflater = LayoutInflater.from(context);
    items = it;
    paths = pa;
    mIcon1 = BitmapFactory.decodeResource(context.getResources(),
                                          R.drawable.back01);
    mIcon2 = BitmapFactory.decodeResource(context.getResources(),
                                          R.drawable.back02);
    mIcon3 = BitmapFactory.decodeResource(context.getResources(),
                                          R.drawable.folder);
    mIcon4 = BitmapFactory.decodeResource(context.getResources(),
                                          R.drawable.doc);
  }
```

③ 使用自定义的 file_row 作为 Layout 布局样式，然后分别设置“回到根目录”的文字与 icon 和“回到上一层”的文字与 icon，具体代码如下所示。

```
  @Override
  public View getView(int position,View convertView,ViewGroup parent)
  {
    ViewHolder holder;

    if(convertView == null)
    {
      /*使用自定义的 file_row 作为 Layout*/
```

```
            convertView = mInflater.inflate(R.layout.file_row, null);
            /*初始化 holder 的 text 与 icon*/
            holder = new ViewHolder();
            holder.text = (TextView) convertView.findViewById(R.id.text);
            holder.icon = (ImageView) convertView.findViewById(R.id.icon);

            convertView.setTag(holder);
        }
        else
        {
            holder = (ViewHolder) convertView.getTag();
        }
        File f=new File(paths.get(position).toString());
        /*设置"回到根目录"的文字与 icon*/
        if(items.get(position).toString().equals("b1"))
        {
            holder.text.setText("Back to / ");
            holder.icon.setImageBitmap(mIcon1);
        }
        /*设置"回到上一层"的文字与 icon*/
        else if(items.get(position).toString().equals("b2"))
        {
            holder.text.setText("Back to ...");
            holder.icon.setImageBitmap(mIcon2);
        }
        /*设置"文件或文件夹"的文字与 icon*/
        else
        {
            holder.text.setText(f.getName());
            if(f.isDirectory())
            {
                holder.icon.setImageBitmap(mIcon3);
            }
            else
            {
                holder.icon.setImageBitmap(mIcon4);
            }
        }
        return convertView;
    }
    /*class ViewHolder*/
    private class ViewHolder
    {
        TextView text;
        ImageView icon;
    }
}
```

执行后会显示系统中的文件，当选择一个文件夹后会继续显示此文件中的内容，如图 4-28 所示。

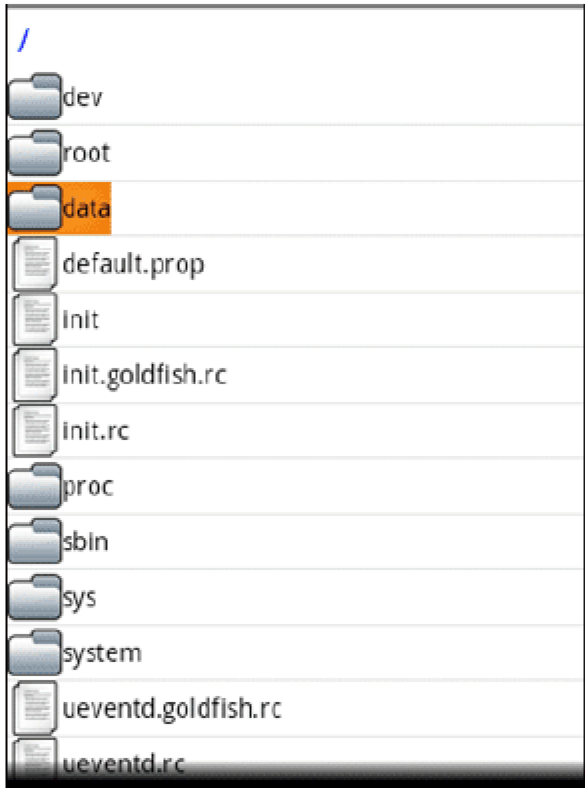


图 4-28　执行效果

4.15　清除、还原手机桌面

实例 069	清除、还原手机桌面
源码路径	光盘:\daima\069
视频路径	光盘:\视频\069
实例必备	069.Android 本地广播.pdf

4.15.1　实例说明

在 Android 手机系统有一个默认的开机主界面，如图 4-29 所示。

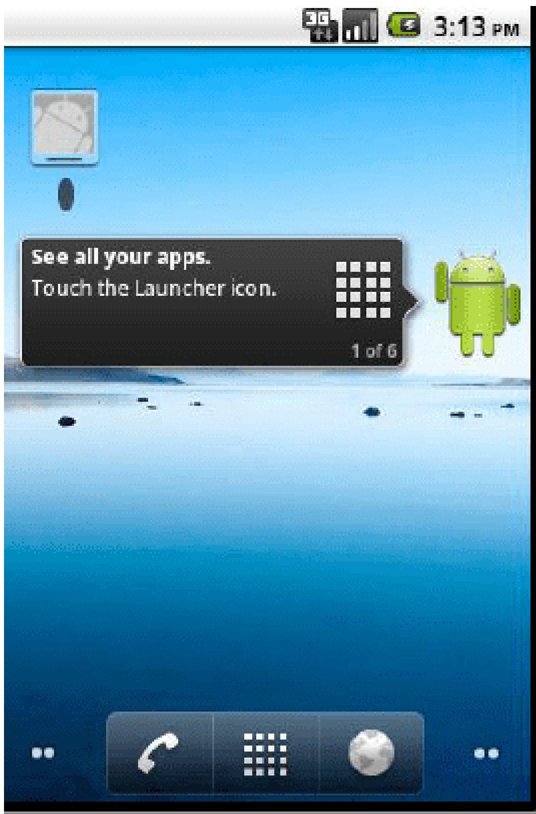


图 4-29　默认主界面

可以通过编程的方法控制默认主界面显示的内容。在本实例中，首先在文件 AndroidManifest.xml 中设置操作权限，然后重写 ContextWrapper 的方法 clearWallpaper()，这样就可以设置操作默认主界面的显示内容。

4.15.2 具体实现

编写主程序文件，此文件的具体实现流程如下所示。

（1）单击 Button 控件后用 OnClickListener 启动事件，具体代码如下所示。

```
    private Button mButton1;
    /** Called when the activity is first created.*/
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mButton1 =(Button) findViewById(R.id.myButton1);

    /*设置 Button 用 OnClickListener 来启动事件*/
    mButton1.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        // TODO Auto-generated catch block
        try
        {
          /*清除背景图案*/
          clearWallpaper();
          Toast.makeText(example.this, getString(R.string.str_done)
                ,Toast.LENGTH_SHORT).show();
        }
        catch (IOException e)
        {
          e.printStackTrace();
        }
      }
    });
  }
```

（2）重写方法 clearWallpaper()，用于清除当前设置的桌面，从而还原成原来的默认设置，具体代码如下所示。

```
  @Override
  public void clearWallpaper() throws IOException
  {
    // TODO Auto-generated method stub
    super.clearWallpaper();
  }
}
```

最后需要在文件 AndroidManifest.xml 中设置 SET_WALLPAPER 权限，具体代码如下所示。

```
    <uses-permission android:name="android.permission.SET_WALLPAPER" />
</manifest>
```

执行代码后将显示一个“删除”按钮，单击此按钮后会还原为默认的背景界面。

4.16　修改手机屏幕的显示方向

实例 070	修改手机屏幕的显示方向
源码路径	光盘:\daima\070
视频路径	光盘:\视频\070
实例必备	070.拨打电话.pdf

4.16.1　实例说明

广大读者对屏幕旋转肯定不会陌生，很多智能手机都能根据拿手机的方式动态横向或纵向显示屏幕中的内容。在本实例的屏幕中插入了一个按钮，单击按钮后会先判断当前屏幕的方向，即如果是横向显示则改为纵向显示，如果是纵向显示则改为横向显示。在具体实现上，可用方法 setRequestedOrientation()来改变屏幕方向，如果要获取当前的屏幕方向，则需要访问方法 getRequestedOrientation()。

4.16.2　具体实现

（1）编写主程序文件，此文件的具体实现流程如下所示。

① 判断 getRequestedOrientation()的值是否为-1，如果是-1，则表示在 Activity 属性中没有设置 Android:screenOrientation 的值，即表示即使单击了按钮，也无法判断出屏幕的方向，不会实现屏幕方向的更改。具体代码如下所示。

```
private TextView mTextView01;
private Button mButton01;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mButton01 = (Button)findViewById(R.id.myButton1);
  mTextView01 = (TextView)findViewById(R.id.myTextView1);

  if(getRequestedOrientation()==-1)
  {
    mTextView01.setText(getResources().getText
    (R.string.str_err_1001));
  }
```

② 定义方法 setOnClickListener(new Button.OnClickListener()，当用户单击按钮后开始旋转屏幕画面，具体代码如下所示。

```
  mButton01.setOnClickListener(new Button.OnClickListener()
  {
    @Override
```

```
        public void onClick(View arg0)
        {
          /*方法一：重写 getRequestedOrientation*/

          /*若无法取得 screenOrientation 属性*/
          if(getRequestedOrientation()==-1)
          {
            /*提示无法进行画面旋转功能，因无法判别 Orientation*/
            mTextView01.setText(getResources().getText
            (R.string.str_err_1001));
          }
          else
          {
            if(getRequestedOrientation()==
               ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE)
            {
              /*若当下为横向，则更改为竖向呈现*/
              setRequestedOrientation
              (ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
            }
            else if(getRequestedOrientation()==
                    ActivityInfo.SCREEN_ORIENTATION_PORTRAIT)
            {
              /*若当下为竖向，则更改为横向呈现*/
              setRequestedOrientation
              (ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
            }
          }
        }
      });
    }
```

③ 定义方法 setRequestedOrientation(int requestedOrientation)，判断当前要更改的屏幕方向，并实现旋转处理，具体代码如下所示。

```
    @Override
    public void setRequestedOrientation(int requestedOrientation)
    {
      /*判断要更改的方向，以 Toast 提示*/
      switch(requestedOrientation)
      {
        /*更改为 LANDSCAPE*/
        case (ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE):
          mMakeTextToast
          (
            getResources().getText(R.string.str_msg1).toString(),
            false
          );
          break;
        /*更改为 PORTRAIT*/
        case (ActivityInfo.SCREEN_ORIENTATION_PORTRAIT):
          mMakeTextToast
          (
            getResources().getText(R.string.str_msg2).toString(),
```

```
                false
            );
            break;
        }
        super.setRequestedOrientation(requestedOrientation);
    }
```

④ 定义方法 getRequestedOrientation(int requestedOrientation)，获取当前屏幕方向，并通过方法 mMakeTextToast()输出提示信息，具体代码如下所示。

```
    @Override
    public int getRequestedOrientation()
    {
        // TODO Auto-generated method stub

        /*重写 getRequestedOrientation()方法，可获取当前屏幕的方向*/
        return super.getRequestedOrientation();
    }

    public void mMakeTextToast(String str, boolean isLong)
    {
        if(isLong==true)
        {
            Toast.makeText(example.this, str, Toast.LENGTH_LONG).show();
        }
        else
        {
            Toast.makeText(example.this, str, Toast.LENGTH_SHORT).show();
        }
    }
}
```

（2）编写文件 AndroidManifest.xml，在其中设置 Activity 的 Android:screenOrientation 属性，具体代码如下所示。

```
        <activity
            android:name=".example106"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
>
```

执行后在屏幕中显示一个按钮，如图 4-30 所示。单击“改变方向”按钮后会旋转屏幕，如图 4-31 所示。

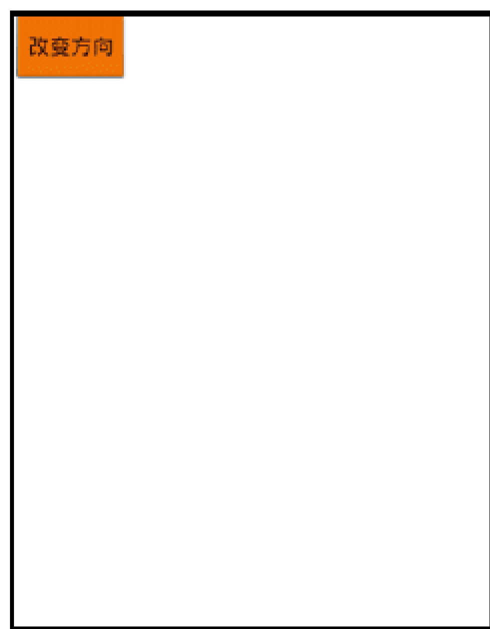


图 4-30　初始效果



图 4-31　实现旋转

第 5 章　自动化服务应用实战

收到了一条短信后，立即在屏幕中弹出一个提示框“您有一条新的信息”，相信读者应该有上述经历。手机是一个神奇的工具，在有新情况时会自动发出提示信息。这种自动化提示信息功能是通过编程的方式实现的。在本章的内容中，将通过具体的实例来讲解在 Android 系统中实现自动化服务的基本用法。

5.1　获取当前运行程序的路径

实例 071	获取当前运行程序的路径
源码路径	光盘:\daima\071
视频路径	光盘:\视频\071
实例必备	071.获取手机屏幕的分辨率.pdf ① 实例说明 ② 具体实现 ③ 特别提醒——一个模拟器模拟短信操作

5.1.1　实例说明

在运行 Android 程序后，可以通过编程的方式来获取正在运行程序的路径，此路径通常位于\data\data\package name 目录下，此处的 package name 是程序的 pakeage 名称。在本实例中设计了两个按钮，会分别触发获取 File 和 Cache 路径的事件。当用户选择某文件后会弹出一个新页面，此新页面以 ListActivity 来显示其目录或文件。当打开目录时，还可以看到该目录下的文件。

5.1.2　具体实现

（1）编写主程序文件，其具体实现流程如下所示。

① 定义两个按钮变量 myButton1 和 myButton2，然后定义两个文件变量 cacheDir 和 fileDir。具体代码如下所示。

```
private Button myButton1;
private Button myButton2;
private File cacheDir;
private File fileDir;
```

② 根据单击的按钮获取目前 Cache 目录和 File 目录，具体代码如下所示。

```
@Override
public void onCreate(Bundle savedInstanceState)
```

```
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    myButton1 = (Button) findViewById(R.id.myButton1);
    myButton2 = (Button) findViewById(R.id.myButton2);

    /* 取得目前 Cache 目录*/
    cacheDir = this.getCacheDir();
    /* 取得目前 File 目录*/
    fileDir = this.getFilesDir();
```

③ 定义单击两个按钮的处理事件 myButton1.setOnClickListener 和 myButton2.setOnClickListener，具体代码如下所示。

```
    myButton1.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        String path = fileDir.getParent() + java.io.File.separator
                + fileDir.getName();
        showListActivity(path);
      }
    });
    myButton2.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        String path = cacheDir.getParent() + java.io.File.separator
                + cacheDir.getName();
        showListActivity(path);
      }
    });
  }
```

④ 调用 example_1 中定义的功能，然后传入获取路径，具体代码如下所示。

```
  private void showListActivity(String path)
  {
    Intent intent = new Intent();
    intent.setClass(example13.this, example_1.class);
    Bundle bundle = new Bundle();
    bundle.putString("path", path);
    intent.putExtras(bundle);
    startActivity(intent);
  }

}
```

（2）编写文件 example_1.java，其具体实现流程如下所示。

① 定义类 example_1，此类继承了 ListActivity。具体代码如下所示。

```
public class example_1 extends ListActivity
{
```

```
    private List<String> items = null;
    private String path;
    /** Called when the activity is first created.*/
    public void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.mylist);
      Bundle bunde = this.getIntent().getExtras();
      /*取得 example103 所传的路径*/
      path = bunde.getString("path");
      this.setTitle(path);
      java.io.File file = new java.io.File(path);
      /*列出该路径下的所有文件*/
      fill(file.listFiles());
    }
```

② 定义方法 onListItemClick()用于显示传入的文件和文件目录，具体代码如下所示。

```
    @Override
    protected void onListItemClick
    (ListView l, View v, int position, long id)
    {
      File file = new File
      (path + java.io.File.separator + items.get(position));

      if (file.isDirectory())
      {
        fill(file.listFiles());
      }
    }
```

③ 定义方法 fill(File[ ] files)将取得的文件名放入到 ArrayList 中，具体代码如下所示。

```
    private void fill(File[ ] files)
    {
      items = new ArrayList<String>();
      if (files == null)
      {
        return;
      }
      /*将取得的文件名放入 ArrayList*/
      for (File file : files)
      {
        items.add(file.getName());
      }
      /*将 ArrayList 放入 ArrayAdapter*/
      ArrayAdapter<String> fileList = new ArrayAdapter<String>
      (this, android.R.layout.simple_list_item_1, items);
      setListAdapter(fileList);
    }
}
```

（3）编写文件 AndroidManifest.xml，在其中设置两个 Activity，一个在 LAUNCHER 启动时运行，另外一个为取得文件夹信息时所需要显示的 Activity，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="irdc.example103"
        android:versionCode="1"
        android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".example103"
                    android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="irdc.example103.example103_1"></activity>
    </application>
</manifest>
```

执行后在屏幕中显示两个按钮，如图 5-1 所示，当单击一个按钮后会显示对应的目录信息。



图 5-1　执行效果

5.2　获取手机内 SIM 卡的信息

实例 072	获取手机内 SIM 卡的信息
源码路径	光盘:\daima\072
视频路径	光盘:\视频\072
实例必备	072.SIM 卡的意义.pdf

5.2.1　实例说明

在 Android 手机应用中，经常需要获取 SIM 卡内的信息。本实例通过 Android API 中的 TelephonyManager（Android.telephony.TelephonyManager）对象中的方法来读取 SIM 卡的信息。另外，TelephonyManager 还能获取手机的号码，如下面的代码所示。

```
TelephonyManager tm = (TelephonyManager)this.getSystemService(Context.TELEPHONY_SERVICE);
numberText.setText(tm.getLine1Number());
```

其中，加粗部分能够获取本机号码，除此之外，在类 TelephonyManager 中还提供了多种获取手机信息的函数，如 IMEI、IMSI 等，如下面的代码所示。

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
        numberText = (TextView) findViewById(R.id.numberText);
        imeiText = (TextView) findViewById(R.id.imeiText);
        onText = (TextView) findViewById(R.id.onText);
        snText = (TextView) findViewById(R.id.snText);
        imsiText = (TextView) findViewById(R.id.imsiText);
        ssText = (TextView) findViewById(R.id.ssText);
        ntText = (TextView) findViewById(R.id.ntText);
        TelephonyManager tm = (TelephonyManager)this.getSystemService(Context.TELEPHONY_SERVICE);
        numberText.setText(tm.getLine1Number());
        imeiText.setText(tm.getDeviceId());
        onText.setText(tm.getNetworkOperatorName());
        snText.setText(tm.getSimSerialNumber());
        imsiText.setText(tm.getSubscriberId());
        ssText.setText(tm.getNetworkCountryIso());
        ntText.setText(tm.getNetworkOperator());
    }
}
```

通过上述代码中的函数，分别获取了手机号码、IMEI、运营商名称、SIM 卡序列号、IMSI、SIM 卡所在国家（ISO）、运营商编号，运行效果如图 5-2 所示。

手机号码： 15555218135
IMEI： 000000000000000
运营商： Android
sim卡序列号： 89014103211118510720
IMSI： 310260000000000
ISO： us
运营商编号： 310260

图 5-2　运行效果

5.2.2　具体实现

（1）编写主程序文件，此文件的具体实现流程如下所示。

① 载入 main.xml 布局文件，通过方法 add(getResources().getText(R.string.str_list0).toString())将取得的信息写入 List 中，最后通过 if 语句来设置 SIM 卡的状态，具体代码如下所示。

```
private TelephonyManager telMgr;
private List<String> item=new ArrayList<String>();
private List<String> value=new ArrayList<String>();

@SuppressWarnings("static-access")
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  /*载入 main.xml Layout*/
  setContentView(R.layout.main);
  telMgr = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);

  /*将取得的信息写入 List 中*/
  /*取得 SIM 卡状态*/
  item.add(getResources().getText(R.string.str_list0).toString());
  if(telMgr.getSimState()==telMgr.SIM_STATE_READY)
  {
    value.add("良好");
  }
  else if(telMgr.getSimState()==telMgr.SIM_STATE_ABSENT)
```

```
{
    value.add("无 SIM 卡");
}
else
{
    value.add("SIM 卡被锁定或未知的状态");
}
```

② 分别获取 SIM 卡的卡号、供货商代码、供货商名称和国别信息，然后使用自定义的 MyAdapter 将数据传入 ListActivity，具体代码如下所示。

```
/*取得 SIM 卡卡号*/
item.add(getResources().getText(R.string.str_list1).toString());
if(telMgr.getSimSerialNumber()!=null)
{
    value.add(telMgr.getSimSerialNumber());
}
else
{
    value.add("无法取得");
}
/*取得 SIM 卡供货商代码*/
item.add(getResources().getText(R.string.str_list2).toString());
if(telMgr.getSimOperator().equals(""))
{
    value.add("无法取得");
}
else
{
    value.add(telMgr.getSimOperator());
}
/*取得 SIM 卡供货商名称*/
item.add(getResources().getText(R.string.str_list3).toString());
if(telMgr.getSimOperatorName().equals(""))
{
    value.add("无法取得");
}
else
{
    value.add(telMgr.getSimOperatorName());
}
/*取得 SIM 卡国别*/
item.add(getResources().getText(R.string.str_list4).toString());
if(telMgr.getSimCountryIso().equals(""))
{
    value.add("无法取得");
}
else
{
    value.add(telMgr.getSimCountryIso());
}
/*使用自定义的 MyAdapter，将数据传入 ListActivity*/
```

```
        setListAdapter(new MyAdapter(this,item,value));
    }
```

（2）编写文件 MyAdapter.java，其具体实现流程如下所示。

① 声明变量 mInflater、items 和 values，然后定义 MyAdapter 构造器并传入 3 个参数，具体代码如下所示。

```
/*自定义的 Adapter，继承 android.widget.BaseAdapter*/
public class MyAdapter extends BaseAdapter
{
  /*变量声明*/
  private LayoutInflater mInflater;
  private List<String> items;
  private List<String> values;
  /*MyAdapter 的构造器，传入 3 个参数 */
  public MyAdapter(Context context,List<String> item,
                     List<String> value)
  {
    /*参数初始化*/
    mInflater = LayoutInflater.from(context);
    items = item;
    values = value;
  }
```

② 分别覆盖方法 getCount()、getItem(int position)、getItemId(int position)、getView(int position,View convertView,ViewGroup par)，具体代码如下所示。

```
  /*因继承 BaseAdapter，需覆盖以下方法*/
  @Override
  public int getCount()
  {
    return items.size();
  }
  @Override
  public Object getItem(int position)
  {
    return items.get(position);
  }

  @Override
  public long getItemId(int position)
  {
    return position;
  }

  @Override
  public View getView(int position,View convertView,ViewGroup par)
  {
    ViewHolder holder;
    if(convertView == null)
    {
      /*使用自定义的 file_row 作为 Layout*/
      convertView = mInflater.inflate(R.layout.row_layout,null);
```

```
            /*初始化 holder 的 text 与 icon*/
            holder = new ViewHolder();
            holder.text1=(TextView)convertView.findViewById(R.id.myText1);
            holder.text2=(TextView)convertView.findViewById(R.id.myText2);
            convertView.setTag(holder);
        }
        else
        {
            holder = (ViewHolder) convertView.getTag();
        }
        /*设置要显示的信息*/
        holder.text1.setText(items.get(position).toString());
        holder.text2.setText(values.get(position).toString());
        return convertView;
    }
    private class ViewHolder
    {
        /*text1：信息名称
         * text2：信息内容*/
        TextView text1;
        TextView text2;
    }
}
```

（3）编写文件 AndroidManifest.xml，在其中声明读取电话状态的权限，具体代码如下所示。

```
<!-- 设置 READ_PHONE_STATE 权限 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE">
</uses-permission>
```

执行后会显示对应的获取信息，如图 5-3 所示。



图 5-3　执行效果

5.3　查看当前系统中正在运行的程序

实例 073	查看当前系统中正在运行的程序
源码路径	光盘:\daima\073
视频路径	光盘:\视频\073
实例必备	073.AndroidManifest.xml 中的权限.pdf

5.3.1 实例说明

计算机中的任务管理器各位读者肯定不会陌生，其中的进程管理器能够显示当前正在运行的程序。在本实例中插入了一个按钮，单击按钮后会显示当前系统中正在运行的程序。当前运行程序是通过 ActivityManager.getRunningTasks 获取的，然后通过 ListView 将获取的信息显示出来。

当单击按钮后，如果在 ListView 的工作已经结束或被操作系统回收，是不会更新运行列表的。另外，如果不具有访问其他运行程序的权限，也不会显示在 ListView 列表中。

注意：为了保证 Android 的运行，限制了获取运行程序的数量，在本实例中设置了最多获取 30 个进程。

5.3.2 具体实现

（1）编写主程序文件，其具体实现流程如下所示。

① 设置类成员最多能够获取 30 个 Task 数量，具体实现代码如下所示。

```
/*类成员设置取回的最多的 Task 数量*/
private int intGetTastCounter=30;
```

② 设置类成员 ActivityManager 的对象，当单击按钮后获取正在后台运行的工作程序，具体代码如下所示。

```
/*类成员 ActivityManager 对象*/
private ActivityManager mActivityManager;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mButton01 = (Button)findViewById(R.id.myButton1);
  mListView01 = (ListView)findViewById(R.id.myListView1);

  /*单击按钮获取正在后台运行的工作程序*/
  mButton01.setOnClickListener(new Button.OnClickListener()
  {
    @Override
    public void onClick(View v)
    {
      // TODO Auto-generated method stub
      try
      {
        /*ActivityManager 对象向系统获取 ACTIVITY_SERVICE*/
        mActivityManager = (ActivityManager)
        example.this.getSystemService(ACTIVITY_SERVICE);

        arylistTask = new ArrayList<String>();
```

```
            /*以 getRunningTasks()方法获取正在运行中的程序 TaskInfo*/
            List<ActivityManager.RunningTaskInfo> mRunningTasks =
            mActivityManager.getRunningTasks(intGetTastCounter);

            int i = 1;
            /*以循环及 baseActivity 方式获取工作名称与 ID*/
            for (ActivityManager.RunningTaskInfo amTask : mRunningTasks)
            {
              /*baseActivity.getClassName 获取运行进程名称*/
              arylistTask.add("" + (i++) + ": "+
              amTask.baseActivity.getClassName()+
              "(ID=" + amTask.id +")");
            }
            aryAdapter1 = new ArrayAdapter<String>
            (example17.this, R.layout.simple_list_item_1, arylistTask);

            if(aryAdapter1.getCount()==0)
            {
              /*如果没有任何运行的进程，则提示信息*/
              mMakeTextToast
              (
                getResources().getText
                (R.string.str_err_no_running_task).toString(),
                true
              );
            }
            else
            {
              /*发现后台运行的程序，以 ListView Widget 条列呈现*/
              mListView01.setAdapter(aryAdapter1);
            }
          }
          catch(SecurityException e)
          {
            /*当无 GET_TASKS 权限（SecurityException 异常）时提示信息*/
            mMakeTextToast
            (
              getResources().getText
              (R.string.str_err_permission).toString(),
              true
            );
          }
        }
    });
```

③ 监听用户选择某一个正在运行进程时的事件，具体代码如下所示。

```
    mListView01.setOnItemSelectedListener
    (new ListView.OnItemSelectedListener()
```

```
    {
        @Override
        public void onItemSelected
        (AdapterView<?> parent, View v, int id, long arg3)
        {
            // TODO Auto-generated method stub
            /*由于将运行进程以数组存放，所以使用 id 取出数组元素名称*/
            mMakeTextToast(arylistTask.get(id).toString(),false);
        }
        @Override
        public void onNothingSelected(AdapterView<?> arg0)
        {
            // TODO Auto-generated method stub

        }
    });
```

④ 设置当用户选择某一个正在运行进程时的事件处理，具体代码如下所示。

```
    /*当 User 在运行进程上单击时的事件处理*/
    mListView01.setOnItemClickListener
    (new ListView.OnItemClickListener()
    {
        @Override
        public void onItemClick
        (AdapterView<?> parent, View v, int id,   long arg3)
        {
            // TODO Auto-generated method stub
            /*由于将运行进程以数组存放，故以 id 获取数组元素名称*/
            mMakeTextToast(arylistTask.get(id).toString(), false);
        }
    });
  }
```

⑤ 定义方法 mMakeTextToast(String str, boolean isLong)实现一个提醒效果，具体代码如下所示。

```
  public void mMakeTextToast(String str, boolean isLong)
  {
    if(isLong==true)
    {
      Toast.makeText(example.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
      Toast.makeText(example.this, str, Toast.LENGTH_SHORT).show();
    }
  }
}
```

（2）编写文件 AndroidManifest.xml，在此文件中声明 GET_TASKS 权限，具体代码如下所示。

```
<uses-permission android:name="android.permission.GET_TASKS"></uses-permission>
```

执行后在屏幕中显示“获取运行的程序”按钮，单击按钮后列表显示当前正在运行的程序，如图 5-4 所示。



图 5-4　当前运行程序

5.4　收到短信后自动发送提示信息

实例 074	收到短信后自动发送提示信息
源码路径	光盘:\daima\074
视频路径	光盘:\视频\074
实例必备	074.一个模拟器模拟短信操作.pdf

5.4.1　实例说明

在手机系统中设置广播系统 BroadcastReseiver，用于实时监听手机内的短信状况。当手机收到短信后，会通过 Notification 在状态栏中显示短信的摘要信息。本实例设计了这样一个程序：当用户收到短信后，系统将自动在手机屏幕上显示“有短信”的提示。在实例中会将接收到的短信对象解析为可以识别发信人号码和短信正文的字符串。本实例的难点是如何向系统注册一个常驻的 BroadcastReseiver 对象，然后在后台中聆听短信事件，最后将短信内容编译出来。

5.4.2　具体实现

（1）编写主程序文件，目的是使用 TextView 文字显示“等待接收信息中...”的提示，具体代码如下所示。

```
private TextView mTextView1;
/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*通过 findViewById 构造器创建 TextView 对象*/
        mTextView1 = (TextView) findViewById(R.id.myTextView1);
        mTextView1.setText("等待接收信息中...");
    }
}
```

（2）编写文件 example_SMS.java，此文件的具体实现流程如下所示。

① 引用 BroadcastReceiver 类，使用 telephoney.gsm.SmsMessage 收取短信，并使用 Toast 类通知用户收到短信，具体代码如下所示。

```
/*必须引用 BroadcastReceiver 类*/
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
/*必须引用 telephoney.gsm.SmsMessage 来收取短信*/
import android.telephony.gsm.SmsMessage;
/*必须引用 Toast 类来告知用户收到短信*/
import android.widget.Toast;
```

② 自定义继承于 BroadcastReceiver 的类 example_SMS，此类用于聆听系统服务广播的信息，具体代码如下所示。

```
  /*声明静态字符串，并使用 android.provider.Telephony.SMS_RECEIVED 作为 Action 为短信的依据*/
  private static final String mACTION =
  "android.provider.Telephony.SMS_RECEIVED";

  @Override
  public void onReceive(Context context, Intent intent)
  {
    // TODO Auto-generated method stub
    /*判断传来的 Intent 是否为短信*/
    if (intent.getAction().equals(mACTION))
    {
      /*建构一字符串集合变量 sb*/
      StringBuilder sb = new StringBuilder();
      /*接收由 Intent 传来的数据*/
      Bundle bundle = intent.getExtras();
      /*判断 Intent 是否有数据*/
      if (bundle != null)
      {
        /*pdus 为  android 内置短信参数  identifier
         * 通过 bundle.get("")返回一包含 pdus 的对象*/
        Object[ ] myOBJpdus = (Object[ ]) bundle.get("pdus");
        /*构建短信对象 array，并依据收到的对象长度来创建 array 的大小*/
        SmsMessage[ ] messages = new SmsMessage[myOBJpdus.length];
        for (int i = 0; i<myOBJpdus.length; i++)
        {
          messages[i] =
```

```
                SmsMessage.createFromPdu((byte[ ]) myOBJpdus[i]);
            }

            /*将送来的短信合并自定义信息于 StringBuilder 当中*/
            for (SmsMessage currentMessage : messages)
            {
                sb.append("正在接收到来自:\n");
                /*发来信息者的电话号码*/
                sb.append(currentMessage.getDisplayOriginatingAddress());
                sb.append("\n------发来的短信------\n");
                /*取得传来信息的 BODY*/
                sb.append(currentMessage.getDisplayMessageBody());
            }
        }
```

上述代码的实现流程：声明静态字符串，并使用 android.provider.Telephony.SMS_RECEIVED 作为 Action 为短信的依据；用 if 语句判断传来的 Intent 是否为短信，如果是则先建构一字符串集合变量 sb，然后接收由 Intent 传来的数据；通过 if 语句判断 Intent 是否有数据信息。

③ 以 Notification(Toase)提醒的方式显示通知信息，返回到主 Activity 让其以一个全新的 task 任务来运行，具体代码如下所示。

```
        Toast.makeText
        (
            context, sb.toString(), Toast.LENGTH_LONG
        ).show();

        /*返回主 Activity*/
        Intent i = new Intent(context, example1.class);
        /*设置让其以一个全新的 task 来运行*/
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
    }
```

（3）编写文件 AndroidManifest.xml，在其中向系统注册常驻的 receiver 接收器，设置此 receiver 的 intent-filter 名为 android.provider.Telephony.SMS_RECEIVED，并声明 permission.RECEIVE_SMS 权限，具体代码如下所示。

```
      <!-- 建立 receiver 来聆听系统广播信息 -->
      <receiver android:name="example_SMS">
      <!--设定要捕捉的信息名称为 provider 中 Telephony.SMS_RECEIVED -->
      <action
        android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
      </receiver>
</application>
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
```

执行后的效果如图 5-5 所示。当接收到短信后会在屏幕中显示对应的提示信息，并且同时在手机中的收件箱显示收到的短信，如图 5-6 所示。



图 5-5　初始效果



图 5-6　短信提示

5.5　获取手机剩余的电池容量

实例 075	获取手机剩余的电池容量
源码路径	光盘:\daima\075
视频路径	光盘:\视频\075
实例必备	075.Receiver 的作用.pdf

5.5.1　实例说明

在选购手机过程中，电池容量是一个十分重要的因素。在使用过程中，最担心的是害怕手机没电而影响业务。为此，显示电池容量的应用变得愈发重要。在本实例中，使用 BroadcastReceiver 的特性来获取手机电池的容量。通过注册 BroadcastReceiver 时设置的 IntentFilter 来获取系统发出的 Intent.ACTION_BATTERY_CHANGED，然后以此来获取电池的容量。

5.5.2　具体实现

编写主程序文件，其具体实现流程如下所示。

（1）声明 3 个变量对象并创建 BroadcastReceiver，如果捕捉到的 action 是 ACTION_BATTERY_CHANGED，则运行 onBatteryInfoReceiver()方法，具体实现代码如下所示。

```
/*变量声明*/
private int intLevel;
private int intScale;
private Button mButton01;

/*创建 BroadcastReceiver*/
private BroadcastReceiver mBatInfoReceiver=new BroadcastReceiver()
{
   public void onReceive(Context context, Intent intent)
```

```
    {
        String action = intent.getAction();
        /*如果捕捉到的 action 是 ACTION_BATTERY_CHANGED，则运行 onBatteryInfoReceiver()*/
        if (Intent.ACTION_BATTERY_CHANGED.equals(action))
        {
            intLevel = intent.getIntExtra("level", 0);
            intScale = intent.getIntExtra("scale", 100);
            onBatteryInfoReceiver(intLevel,intScale);
        }
    }
```

（2）载入主布局文件 main.xml，初始化 Button 控件和设置单击后的动作，注册系统 BroadcastReceiver 广播事件来访问电池容量大小，具体实现代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    /*载入 main.xml Layout*/
    setContentView(R.layout.main);
    /*初始化 Button，并设置单击后的动作*/
    mButton01 = (Button)findViewById(R.id.myButton1);
    mButton01.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            /*注册一个系统 BroadcastReceiver，作为访问电池容量之用*/
            registerReceiver
            (
                mBatInfoReceiver,
                new IntentFilter(Intent.ACTION_BATTERY_CHANGED)
            );
        }
    });
}
```

（3）定义方法 onBatteryInfoReceiver()，首先创建一个背景模糊的 Window 界面，并将对话框放在前景显示，然后将取得的电池容量显示于对话框中，最后设置返回主画面的按钮，具体代码如下所示。

```
/*捕捉到 ACTION_BATTERY_CHANGED 时要运行的方法*/
public void onBatteryInfoReceiver(int intLevel, int intScale)
{
    final Dialog d = new Dialog(example.this);
    d.setTitle(R.string.str_dialog_title);
    d.setContentView(R.layout.mydialog);
    /*创建一个背景模糊的 Window，且将对话框放在前景*/
    Window window = d.getWindow();
    window.setFlags
    (
        WindowManager.LayoutParams.FLAG_BLUR_BEHIND,
        WindowManager.LayoutParams.FLAG_BLUR_BEHIND
    );
    /*将取得的电池容量显示于对话框中*/
    TextView mTextView02=(TextView)d.findViewById(R.id.myTextView2);
```

```
        mTextView02.setText
        (
          getResources().getText(R.string.str_dialog_body)+
          String.valueOf(intLevel * 100 / intScale) + "%"
        );
        /*设置返回主画面的按钮*/
        Button mButton02 = (Button)d.findViewById(R.id.myButton2);
        mButton02.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v)
          {
            /*反注册 Receiver，并关闭对话窗口*/
            unregisterReceiver(mBatInfoReceiver);
            d.dismiss();
          }
        });
        d.show();
    }
}
```

执行后的效果如图 5-7 所示。当单击“点击后获取电池电量”按钮后会显示当前电池的容量，如图 5-8 所示。

演示获取当前电池的容量



图 5-7　执行效果



图 5-8　显示容量

5.6　来电时自动发送提醒信息

实例 076	来电时自动发送提醒信息
源码路径	光盘:\daima\076
视频路径	光盘:\视频\076
实例必备	076.TelephonyManager 和 PhoneStateListener.pdf

5.6.1　实例说明

在当前手机系统中，如果有电话打进来会自动在屏幕中显示来电用户的姓名或电话号码。在 Android 系统中，可以通过 PhoneStateListener 中的方法监听来电状态。在具体实现时，需要创建

PhoneStateListener 对象，重写其中的 onCallStateChanged()方法，并通过传入的 state 来判断来电状态。

要获取来电状态，需要用户读取电话状态的权限，否则不能成功获取状态。需要在模拟器中事先添加一个联系人记录，并为其命名。这样当电话进来后，会在屏幕中显示其名字。如果是非通讯录中的来电，则在屏幕中显示 Unknown Caller。

5.6.2　具体实现

（1）编写主程序文件，其具体实现流程如下所示。

① 引用主布局文件 main.xml，使用 TextView 对象 myTextView1 显示提示信息，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  myTextView1 = (TextView) findViewById(R.id.myTextView1);
```

② 通过 TelephonyManager 对象 tm 来获取电话服务，通过 tm.listen 来监听当前电话的状态，具体代码如下所示。

```
  /*添加自己实现的 PhoneStateListener*/
  exPhoneCallListener myPhoneCallListener =
  new exPhoneCallListener();

  /*取得电话服务*/
  TelephonyManager tm =
  (TelephonyManager) this.getSystemService
  (Context.TELEPHONY_SERVICE);

  /*注册电话通信 Listener*/
  tm.listen
  (
    myPhoneCallListener,
    PhoneStateListener.LISTEN_CALL_STATE
  );
}
```

③ 使用内部类继承 PhoneStateListener，然后重写 onCallStateChanged 电话状态的改变事件，这样，当状态改变时改变 myTextView1 中的文字和颜色，并分别设置无任何状态、接起电话、电话进来的显示，具体代码如下所示。

```
public class exPhoneCallListener extends PhoneStateListener {
  /*重写 onCallStateChanged，
  当状态改变时改变 myTextView1 的文字及颜色*/
  public void onCallStateChanged(int state, String incomingNumber)
  {
    switch (state)
    {
      /*无任何状态时*/
      case TelephonyManager.CALL_STATE_IDLE:
        myTextView1.setTextColor
        (
          getResources().getColor(R.drawable.red)
```

```
            );
            myTextView1.setText("没有电话进来");
            break;
          /*接起电话时*/
          case TelephonyManager.CALL_STATE_OFFHOOK:
            myTextView1.setTextColor
            (
              getResources().getColor(R.drawable.green)
            );
            myTextView1.setText("正在通话中");
            break;
          /*电话进来时*/
          case TelephonyManager.CALL_STATE_RINGING:
            getContactPeople(incomingNumber);
            break;
          default:
            break;
        }
        super.onCallStateChanged(state, incomingNumber);
      }
    }
```

④ 使用方法 getContactPeople()获取手机内的联系人信息，然后在 cursor 中存放“联系人”的字段名称，具体代码如下所示。

```
    private void getContactPeople(String incomingNumber)
    {
      myTextView1.setTextColor(Color.BLUE);
      ContentResolver contentResolver = getContentResolver();
      Cursor cursor = null;
      /*cursor 中要存放的字段名称*/
      String[ ] projection = new String[ ]
      {
        Contacts.People._ID,
        Contacts.People.NAME,
        Contacts.People.NUMBER
      };
```

⑤ 通过来电号码查找对应的联系人，如果查找到则显示此号码对应的姓名，如果没有查找到则只显示号码，具体实现代码如下所示。

```
      /*根据来电电话号码查找该联系人*/
      cursor = contentResolver.query
      (
        Contacts.People.CONTENT_URI, projection,
        Contacts.People.NUMBER + "=?",
        new String[ ]
        {
          incomingNumber
        },
        Contacts.People.DEFAULT_SORT_ORDER
      );
      /*找不到联系人*/
      if (cursor.getCount() == 0)
```

```
    {
        myTextView1.setText("unknown Number:" + incomingNumber);
    }
    else if (cursor.getCount() > 0)
    {
        cursor.moveToFirst();
        /*在 projection 数组中名字放在第 1 个位置*/
        String name = cursor.getString(1);
        myTextView1.setText(name + ":" + incomingNumber);
    }
  }
}
```

（2）编写文件 AndroidManifest.xml，具体代码如下所示。

```
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
```

通过上述代码获取如下两个权限。

☑ android.permission.READ_CONTACTS：读取通讯录权限。

☑ android.permission.READ_PHONE_STATE：获取电话状态权限。

执行后的效果如图 5-9 所示，当打来电话后会显示来电的基本信息，如图 5-10 所示。

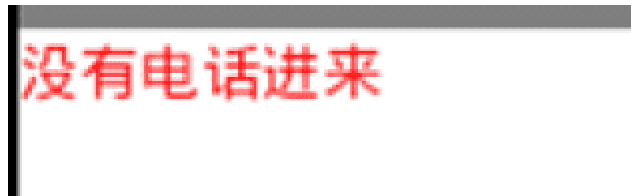


图 5-9　执行效果



图 5-10　来电后界面

5.7　获取手机中存储卡的容量

实例 077	获取手机中存储卡的容量
源码路径	光盘:\daima\077
视频路径	光盘:\视频\077
实例必备	077.使用 FAT32 格式的磁盘镜像作为 SD 卡的模拟.pdf

5.7.1　实例说明

随着手机娱乐功能的增加，需要存储卡功能。存储卡是可以随时插拔的，每次插拔时会对操作系统进行 ACTION broadcast。为了便于下载感兴趣的资源，很有必要及时了解存储卡的容量信息。在本

实例中，使用 StatFs 文件系统来获取 MicroSD 存储卡的剩余容量。在具体实现时，需要先判断是否安装存储卡，如果不存在则不予计算。为了更好地显示容量，在屏幕布局中插入了一个 ProgressBar Widget 控件，这样使显示效果更加一目了然。

5.7.2 具体实现

编写主程序文件，其具体实现流程如下所示。

（1）定义单击按钮监听事件 setOnClickListener，单击后触发事件处理程序，调用方法 showSize() 显示存储卡的剩余容量。具体代码如下所示。

```
    myButton.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        showSize();
      }
    });
  }
```

（2）定义方法 showSize()来显示存储卡的容量大小，具体代码如下所示。

```
  private void showSize() {
    /* 将 TextView 及 ProgressBar 设置为空值及 0*/
    myTextView.setText("");
    myProgressBar.setProgress(0);
    /*判断存储卡是否插入*/
    if (Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED))
    {
      /*取得 SD 卡文件路径，一般是\sdcard*/
      File path = Environment.getExternalStorageDirectory();
      /*StatFs 看文件系统空间使用状况*/
      StatFs statFs = new StatFs(path.getPath());
      /*Block 的 size*/
      long blockSize = statFs.getBlockSize();
      /*总 Block 数量*/
      long totalBlocks = statFs.getBlockCount();
      /*已使用的 Block 数量*/
      long availableBlocks = statFs.getAvailableBlocks();
      String[ ] total = fileSize(totalBlocks * blockSize);
      String[ ] available = fileSize(availableBlocks * blockSize);
      /*getMax 取得在 main.xml 中 ProgressBar 设置的最大值*/
      int ss = Integer.parseInt(available[0]) * myProgressBar.getMax()
          / Integer.parseInt(total[0]);
      myProgressBar.setProgress(ss);
      String text = "总共" + total[0] + total[1] + "\n";
      text += "可用" + available[0] + available[1];
      myTextView.setText(text);
    } else if (Environment.getExternalStorageState().equals(
        Environment.MEDIA_REMOVED))
    {
```

```
            String text = "SD CARD 已删除";
            myTextView.setText(text);
        }
    }
    /*返回为字符串数组[0]的大小，单位为 KB 或 MB*/
    private String[ ] fileSize(long size)
    {
        String str = "";
        if (size >= 1024)
        {
            str = "KB";
            size /= 1024;
            if (size >= 1024)
            {
                str = "MB";
                size /= 1024;
            }
        }
        DecimalFormat formatter = new DecimalFormat();
        /*每 3 个数字用,分隔，如 1,000*/
        formatter.setGroupingSize(3);
        String result[ ] = new String[2];
        result[0] = formatter.format(size);
        result[1] = str;
        return result;
    }
}
```

上述代码的具体实现流程如下所示。

① 分别设置 TextView 和 ProgressBar 为空值和 0。

② 获取 SD 卡文件路径。

③ 通过 StatFs 查看文件系统空间使用状况。

④ 分别获取总 Block 数量和已使用的 Block 数量。

⑤ 通过 getMax 获取在 main.xml 中 ProgressBar 设置的最大值。

⑥ 显示出容量信息。

⑦ 如果没有 SD 卡，则输出“SD CARD 已删除”的提示。

执行后的效果如图 5-11 所示。

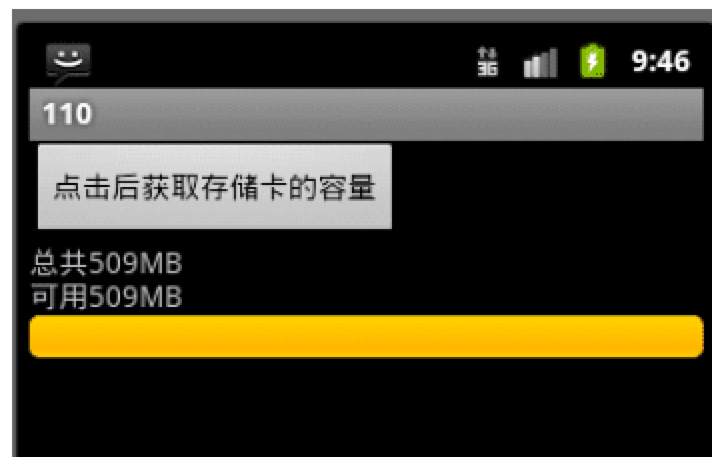


图 5-11　执行效果

在使用 Android 模拟器时，可以使用 FAT32 格式的磁盘镜像作为 SD 卡的模拟，具体实现过程如下所示。

（1）进入 Android SDK 目录下的 tools 子目录，运行如下命令。

```
mksdcard -l sdcard 512M /your_path_for_img/sdcard.img
```

这样就创建了一个 512MB 的 SD 卡镜像文件。

（2）在运行模拟器时指定模拟存储卡路径，在此注意需要使用完整路径。

```
emulator -sdcard /your_path_for_img/sdcard.img
```

此时在模拟器中可以使用/sdcard 这个路径来指向模拟的 SD 卡。

在使用 mksdcard 命令时要注意如下 6 点。

- ☑ mycard 命令可以使用 3 种单位：B、K 和 M。如果只使用数字，表示字节。后面还可以跟 K，如 262144K，即 256M。
- ☑ mycard 建立的虚拟文件最小为 8MB，也就是说，模拟器只支持大于 8MB 的虚拟文件。
- ☑ -l 命令行参数表示虚拟磁盘的卷标，可以没有该参数。
- ☑ 虚拟文件的扩展名可以是任意的，如 mycard.abc。
- ☑ mksdcard 命令不会自动建立不存在的目录，因此，在执行上面命令之前，要先在当前目录中建立一个 card 目录。
- ☑ mksdcard 命令是按实际大小生成的 sdcard 虚拟文件。也就是说，生成 256MB 的虚拟文件的尺寸就是 256M，如果生成较大的虚拟文件，要查看是否有充足的硬盘空间。

在执行完上面的命令后，下面的命令可以启动 Android 模拟器。

```
emulator -avd avd1 -sdcard card/mycard.img
```

如果在开发环境（Eclipse）中，可以在 Run Configuration 对话框中设置启动参数，当然，也可以在 Preferences 对话框中设置默认启动参数。这样在新建立的 Android 工程中就自动加入了装载 sdcard 虚拟文件的命令行参数。

如果读者使用 OPhone 虚拟机，设置的方法也是完全一样的，然后在虚拟机的 Setting 中查看是否存在 sdcard。如何查看 sdcard 虚拟设备中的内容呢？方法很多，最简单的就是使用 Android Eclipse 插件带有的 DDMS 透视图来实现。

5.8 管理存储卡和内存卡中的信息

实例 078	管理存储卡和内存卡中的信息
源码路径	光盘:\daima\078
视频路径	光盘:\视频\078
实例必备	078.管理 SD 卡中的内容.pdf

5.8.1 实例说明

在本实例中将添加两个按钮，分别用于添加和删除内存或存储卡内的文件，并且在实例中使用了 3 个 Activity，主程序界面是 Entry Activity，另外两个分别用于处理内存卡和存储卡。当用户选择内存或存储卡后，以列表形式显示其中所有的目录和文件名，并在 MENU 菜单中显示“添加”或“删除”按钮。单击“添加”按钮后会显示一个添加菜单，实现添加文件功能。当单击“删除”按钮后，可以删除指定的文件。

5.8.2 具体实现

（1）编写主程序文件，此文件的具体实现流程如下所示。

① 使用 getFilesDir()方法取得 SD 卡的记录，设置当 SD 卡没有插入时按钮 myButton2 处于不能使用状态，具体代码如下所示。

```
    /*取得目前 File 目录*/
    fileDir = this.getFilesDir();
    /*取得 SD 卡目录*/
    sdcardDir = Environment.getExternalStorageDirectory();
    /*当没有 SD 卡插入时将 myButton2 设为不能按*/
    if (Environment.getExternalStorageState().equals(Environment.MEDIA_REMOVED))
    {
      myButton2.setEnabled(false);
    }
```

② 分别定义按钮单击处理事件 setOnClickListener 和 setOnClickListener，具体代码如下所示。

```
    myButton1.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        String path = fileDir.getParent() + java.io.File.separator
            + fileDir.getName();
        showListActivity(path);
      }
    });
    myButton2.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        String path = sdcardDir.getParent() + sdcardDir.getName();
        showListActivity(path);
      }
    });
  }
```

③ 在方法 showListActivity()中定义一个 Intent 对象 intent，然后将路径传到 example111_1，具体代码如下所示。

```
  private void showListActivity(String path)
  {
    Intent intent = new Intent();
    intent.setClass(example111.this, example8_1.class);
    Bundle bundle = new Bundle();
    /*将路径传到 example111_1*/
    bundle.putString("path", path);
    intent.putExtras(bundle);
    startActivity(intent);
  }
}
```

（2）编写文件 example_1.java，其具体实现流程如下所示。

① 将主 Activity 传来的 path 字符串作为传入路径，如果路径不存在，则使用 java.io.File 来创建。具体代码如下所示。

```
  private List<String> items = null;
  private String path;
  protected final static int MENU_NEW = Menu.FIRST;
```

```
protected final static int MENU_DELETE = Menu.FIRST + 1;

@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.ex111_1);
  Bundle bunde = this.getIntent().getExtras();
  path = bunde.getString("path");
  java.io.File file = new java.io.File(path);
  /*当该目录不存在时创建目录*/
  if (!file.exists())
  {
    file.mkdir();
  }
  fill(file.listFiles());
}
```

② 定义 onOptionsItemSelected，根据用户选择 MENU 选项分别实现添加或删除操作，具体代码如下所示。

```
 public boolean onOptionsItemSelected(MenuItem item)
{
  super.onOptionsItemSelected(item);
  switch (item.getItemId())
  {
    case MENU_NEW:
      /*单击添加 MENU*/
      showListActivity(path, "", "");
      break;
    case MENU_DELETE:
      /*单击删除 MENU*/
      deleteFile();
      break;
  }
  return true;
}
```

③ 定义 onCreateOptionsMenu 来添加需要的 MENU，具体代码如下所示。

```
public boolean onCreateOptionsMenu(Menu menu)
{
  super.onCreateOptionsMenu(menu);
  /*添加 MENU*/
  menu.add(Menu.NONE, MENU_NEW, 0, R.string.strNewMenu);
  menu.add(Menu.NONE, MENU_DELETE, 0, R.string.strDeleteMenu);
  return true;
}
```

④ 当单击某一个文件名时获取此文件的内容，具体代码如下所示。

```
protected void onListItemClick
(ListView l, View v, int position, long id)
{
  File file = new File
  (path + java.io.File.separator + items.get(position));
```

```
        /*单击文件取得文件内容*/
        if (file.isFile())
        {
            String data = "";
            try
            {
                FileInputStream stream = new FileInputStream(file);
                StringBuffer sb = new StringBuffer();
                int c;
                while ((c = stream.read()) != -1)
                {
                    sb.append((char) c);
                }
                stream.close();
                data = sb.toString();
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
            showListActivity(path, file.getName(), data);
        }
    }
```

⑤ 使用方法 fill(File[ ] files)将内容填充到文件，具体代码如下所示。

```
    private void fill(File[ ] files)
    {
        items = new ArrayList<String>();
        if (files == null)
        {
            return;
        }
        for (File file : files)
        {
            items.add(file.getName());
        }
        ArrayAdapter<String> fileList = new ArrayAdapter<String>
        (this,android.R.layout.simple_list_item_1, items);
        setListAdapter(fileList);
    }
```

⑥ 使用方法 showListActivity()显示已经存在的文件列表，主要代码如下所示。

```
    private void showListActivity
    (String path, String ilename, String data)
    {
        Intent intent = new Intent();
        intent.setClass(example_1.this, example_2.class);
        Bundle bundle = new Bundle();
        /*文件路径*/
        bundle.putString("path", path);
        /*文件名*/
        bundle.putString("ilename", ilename);
```

```
        /*文件内容*/
        bundle.putString("data", data);
        intent.putExtras(bundle);

        startActivity(intent);
    }
```

⑦ 使用方法 deleteFile()删除用户选定的文件，具体代码如下所示。

```
    private void deleteFile() {
        int position = this.getSelectedItemPosition();
        if (position >= 0)
        {
            File file = new File(path + java.io.File.separator +
            items.get(position));
            /*删除文件*/
            file.delete();
            items.remove(position);
            getListView().invalidateViews();
        }
    }
}
```

（3）编写文件 example_2.java，其具体实现流程如下。

① 使用 myEditText1 对象放置文件内容，定义 Bundle 对象 bunde 来获取路径 path 和数据 data。具体代码如下所示。

```
     public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ex_2);
        /*放置文件内容的 EditText*/
        myEditText1 = (EditText) findViewById(R.id.myEditText1);

        Bundle bunde = this.getIntent().getExtras();
        path = bunde.getString("path");
        data = bunde.getString("data");
        fileName = bunde.getString("fileName");
        myEditText1.setText(data);
    }
```

② 使用 onOptionsItemSelected 根据用户选择进行操作。当选择 MENU_SAVE 时，保存这个文件。具体代码如下所示。

```
    public boolean onOptionsItemSelected(MenuItem item)
    {
        super.onOptionsItemSelected(item);
        switch (item.getItemId())
        {
            case MENU_SAVE:
                saveFile();
                break;
        }
        return true;
    }
```

③ 使用 onCreateOptionsMenu(Menu menu)添加一个 MENU，具体代码如下所示。

```
public boolean onCreateOptionsMenu(Menu menu)
{
   super.onCreateOptionsMenu(menu);
   /*添加 MENU*/
   menu.add(Menu.NONE, MENU_SAVE, 0, R.string.strSaveMenu);
   return true;
}
```

④ 定义 saveFile()保存文件。先定义 LayoutInflater 对象 factory 以跳出并存档，然后通过 myDialogEditText 获取 Dialog 对话框中的 EditText 数据，最后实现存档处理，具体代码如下所示。

```
private void saveFile()
{
   /*跳出存档的 Dialog*/
   LayoutInflater factory = LayoutInflater.from(this);

   final View textEntryView = factory.inflate
   (R.layout.save_dialog, null);

   Builder mBuilder1 = new AlertDialog.Builder(example8_2.this);

   mBuilder1.setView(textEntryView);
   /*取得 Dialog 中的 EditText*/
   myDialogEditText = (EditText) textEntryView.findViewById
                                              (R.id.myDialogEditText);
   myDialogEditText.setText(fileName);
   mBuilder1.setPositiveButton
   (
      R.string.str_alert_ok,new DialogInterface.OnClickListener()
      {
         public void onClick(DialogInterface dialoginterface, int i)
         {
            /*存档*/
            String Filename = path + java.io.File.separator
                  + myDialogEditText.getText().toString();
            java.io.BufferedWriter bw;
            try
            {
               bw = new java.io.BufferedWriter(new java.io.FileWriter(
                     new java.io.File(Filename)));
               String str = myEditText1.getText().toString();
               bw.write(str, 0, str.length());
               bw.newLine();
               bw.close();
            }
            catch (IOException e)
            {
               e.printStackTrace();
            }
            /*回到 example_1*/
            Intent intent = new Intent();
```

```
                intent.setClass(example_2.this, example8_1.class);
                Bundle bundle = new Bundle();
                /*将路径传到 example_1*/
                bundle.putString("path", path);
                intent.putExtras(bundle);
                startActivity(intent);
                finish();
            }
        });
        mBuilder1.setNegativeButton(R.string.str_alert_cancel, null);
        mBuilder1.show();
    }
}
```

执行后的效果如图 5-12 所示，当单击按钮后会显示对应的存储信息，如图 5-13 所示。当单击 MENU 按钮后会弹出两个控制选项，如图 5-14 所示。此时，可以通过这两个选项分别对存储卡中的数据进行管理。



图 5-12　执行效果

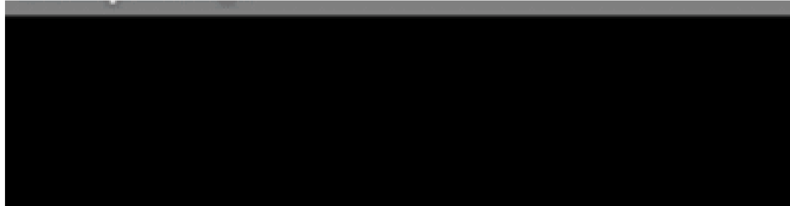


图 5-13　SD 卡的文件信息

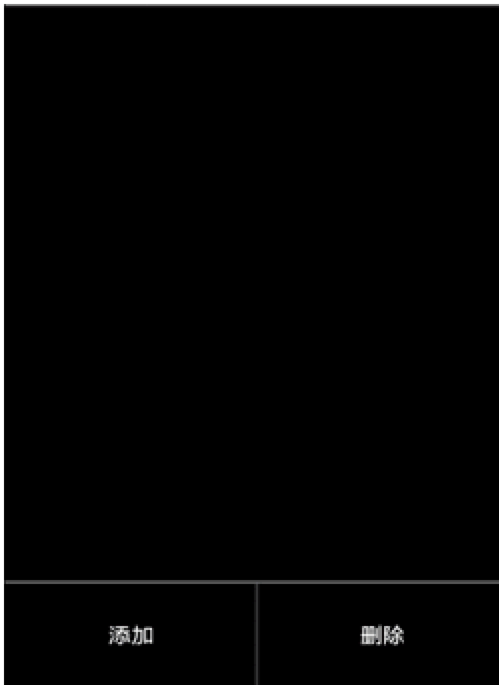


图 5-14　管理 MENU

5.9　设置黑名单来电自动静音

实例 079	设置黑名单来电自动静音
源码路径	光盘:\daima\079
视频路径	光盘:\视频\079
实例必备	079.检测 Android 系统是否静音.pdf

5.9.1　实例说明

几乎在当前的每部手机中都具备黑名单功能，不允许列入黑名单的用户打进电话或发进短信。在本实例中添加一个 EditText，在其中可以输入黑名单用户的电话号码。当此号码来电时，系统会自动将其设置为静音模式。当对方挂机后，系统将自动设置为正常模式，并使用 Toast 提示用户。在具体实现上，通过 setRingerMode 改变铃声模式。在 Android 系统中存在如下 3 种模式。

☑ 正常模式：RINGER_MODE_NORMAL。
☑ 静音模式：RINGER_MODE_SILENT。
☑ 震动模式：RINGER_MODE_VIBRATE。

5.9.2　具体实现

编写主程序文件，其具体实现流程如下所示。

（1）设置 PhoneCallListener 对象 phoneListener，使用 TelephonyManager 获取 Telephony Severice 值，然后查找 TextView 和 EditText 中的数据信息，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*设置 PhoneCallListener*/
  mPhoneCallListener phoneListener=new mPhoneCallListener();
  /*用 TelephonyManager 抓取 Telephony Severice*/
  TelephonyManager telMgr = (TelephonyManager)getSystemService(
      TELEPHONY_SERVICE);
  /*设置 Listen Call*/
  telMgr.listen(phoneListener, mPhoneCallListener.
      LISTEN_CALL_STATE);
  /*获取 TextView 和 EditText 的数据信息*/
  mTextView01 = (TextView)findViewById(R.id.myTextView1);
  mTextView03 = (TextView)findViewById(R.id.myTextView3);
  mEditText1 = (EditText)findViewById(R.id.myEditText1);

}
```

（2）判断 PhoneStateListener 的当前状态，具体代码如下所示。

```
/*判断 PhoneStateListener 当前状态*/
public class mPhoneCallListener extends PhoneStateListener
{
  @Override
  public void onCallStateChanged(int state, String incomingNumber)  {
    switch(state)
    {
      /*获取手机待机状态*/
      case TelephonyManager.CALL_STATE_IDLE:
        mTextView01.setText(R.string.str_CALL_STATE_IDLE);
        try
        {
          AudioManager audioManager = (AudioManager)
                          getSystemService(Context.AUDIO_SERVICE);
          if (audioManager != null)
          {
            /*设置手机为待机时响铃正常*/
            audioManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
            audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

```
            }
        }
        catch(Exception e){
          mTextView01.setText(e.toString());
          e.printStackTrace();
        }
        break;
      /*获取手机状态为通话中*/
      case TelephonyManager.CALL_STATE_OFFHOOK:
        mTextView01.setText(R.string.str_CALL_STATE_OFFHOOK);
        break;
      /*获取的手机状态为来电中*/
      case TelephonyManager.CALL_STATE_RINGING:
        /*显示来电信息*/
        mTextView01.setText(
            getResources().getText(R.string.str_CALL_STATE_RINGING)+
          incomingNumber);
        /*判断输入电话是否一致，一致时用静音*/
        if(incomingNumber.equals(mTextView03.getText().toString())){
        try{
          AudioManager audioManager = (AudioManager)
                          getSystemService(Context.AUDIO_SERVICE);
        if (audioManager != null){
          /*设置响铃为静音*/
          audioManager.setRingerMode(AudioManager.
                RINGER_MODE_SILENT);
          audioManager.getStreamVolume(
                AudioManager.STREAM_RING);
          Toast.makeText(example10.this, getString(R.string.str_msg)
              ,Toast.LENGTH_SHORT).show();
          }
        }
        catch(Exception e)
        {
          mTextView01.setText(e.toString());
          e.printStackTrace();
        break;
        }
      }
    }
    super.onCallStateChanged(state, incomingNumber);
    mEditText1.setOnKeyListener(new EditText.OnKeyListener()
    {
```

（3）通过 onKey 将在 EditText 文本框中输入的数据显示在 TextView 中，具体代码如下所示。

```
      public boolean onKey(View v, int keyCode, KeyEvent event)
      {
        // TODO Auto-generated method stub
        /*将在 EditText 中输入的数据显示在 TextView 中*/
        mTextView03.setText(mEditText1.getText());
        return false;
      }
```

执行后的效果如图 5-15 所示，在输入框中可以输入黑名单号码，如图 5-16 所示。设置完毕后，当此号码来电时会自动设置为静音模式，如图 5-17 所示。



图 5-15　执行效果



图 5-16　黑名单号码



图 5-17　来电静音

注意：上述实例在模拟器中不会显示静音模式图片，但是在真实机器上会显示。

5.10　自动更换手机桌面背景

实例 080	自动更换手机桌面背景
源码路径	光盘:\daima\080
视频路径	光盘:\视频\080
实例必备	080.AlarmManager 的原理.pdf

5.10.1　实例说明

在很多智能手机中，可以设置在不同的时间显示不同屏幕背景。在本实例中，预先准备了 7 张素材图片供用户选择，这些素材图片被保存在 res\drawable 目录下。本实例结合了本书前面介绍的闹钟实例的实现原理，使用 AlarmManage 设置在什么时间执行什么样的动作。

5.10.2　具体实现

（1）编写主程序文件，其具体实现流程如下。

① 声明 7 个 Button 设置按钮、1 个启动按钮和 1 个终止按钮，声明 7 个显示素材图片文件名称的 TextView，具体代码如下所示。

```
/*声明 7 个 Button 设置按钮、1 个启动按钮和 1 个终止按钮*/
private Button mButton1;
private Button mButton2;
private Button setButton1;
private Button setButton2;
private Button setButton3;
```

```
private Button setButton4;
private Button setButton5;
private Button setButton6;
private Button setButton7;
/*声明显示图文件名称的 7 个 TextView*/
private TextView mySet1;
private TextView mySet2;
private TextView mySet3;
private TextView mySet4;
private TextView mySet5;
private TextView mySet6;
private TextView mySet7;
```

② 声明自定义的数据库变量 Dai，用来存放设置的图片地址 Map，具体代码如下所示。

```
/*声明自定义的数据库变量 Dai*/
private Dai db;
/*声明存放设置值的 Map*/
private Map<Integer,Integer> map;
private LayoutInflater inflater;
private int tmpWhich=0;
/*声明存放图文件 id 的数组 bg 与存放图文件名称的数组 bgName*/
private final int[ ] bg =
{R.drawable.b01,R.drawable.b02,R.drawable.b03,R.drawable.b04,
R.drawable.b05,R.drawable.b06,R.drawable.b07};
private final String[ ] bgName =
{"b01.png","b02.png","b03.png","b04.png","b05.png","b06.png",
"b07.png"};
```

③ 将主布局文件载入 main.xml，将在数据库中存放的设置值保存到 map 中，然后初始化各个 TextView 对象，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  /*载入 main.xml 布局文件*/
  setContentView(R.layout.main);
  inflater=(LayoutInflater)getSystemService(
      Context.LAYOUT_INFLATER_SERVICE);
  /*将数据库存放的设置值放入 map 中*/
  initSettingData();
  /*初始化 TextView 对象*/
  mySet1=(TextView) findViewById(R.id.mySet1);
  mySet2=(TextView) findViewById(R.id.mySet2);
  mySet3=(TextView) findViewById(R.id.mySet3);
  mySet4=(TextView) findViewById(R.id.mySet4);
  mySet5=(TextView) findViewById(R.id.mySet5);
  mySet6=(TextView) findViewById(R.id.mySet6);
  mySet7=(TextView) findViewById(R.id.mySet7);
```

④ 根据获取的图像设置显示的图文件名称，实现代码如下所示。

```
  /*设置显示的图文件名称*/
  if(!map.get(0).equals(99))
  {
    mySet1.setText(bgName[map.get(0)]);
  }
```

```
if(!map.get(1).equals(99))
{
  mySet2.setText(bgName[map.get(1)]);
}
if(!map.get(2).equals(99))
{
  mySet3.setText(bgName[map.get(2)]);
}
if(!map.get(3).equals(99))
{
  mySet5.setText(bgName[map.get(3)]);
}
if(!map.get(4).equals(99))
{
  mySet5.setText(bgName[map.get(4)]);
}
if(!map.get(5).equals(99))
{
  mySet6.setText(bgName[map.get(5)]);
}
if(!map.get(6).equals(99))
{
  mySet7.setText(bgName[map.get(6)]);
}
```

⑤ 初始 Button 对象，使用方法 initButton()监听单击不同按钮的事件，具体代码如下所示。

```
/*初始化 Button 对象*/
setButton1=(Button) findViewById(R.id.setButton1);
setButton2=(Button) findViewById(R.id.setButton2);
setButton3=(Button) findViewById(R.id.setButton3);
setButton4=(Button) findViewById(R.id.setButton4);
setButton5=(Button) findViewById(R.id.setButton5);
setButton6=(Button) findViewById(R.id.setButton6);
setButton7=(Button) findViewById(R.id.setButton7);
/*用 initButton()来设置 OnClickListener*/
setButton1=initButton(setButton1,mySet1,0);
setButton2=initButton(setButton2,mySet2,1);
setButton3=initButton(setButton3,mySet3,2);
setButton4=initButton(setButton4,mySet4,3);
setButton5=initButton(setButton5,mySet5,4);
setButton6=initButton(setButton6,mySet6,5);
setButton7=initButton(setButton7,mySet7,6);
```

⑥ 用 setOnClickListener 监听单击启动服务按钮的事件，具体代码如下所示。

```
/*设置启动服务的 Button*/
mButton1=(Button)findViewById(R.id.myButton1);
mButton1.setOnClickListener(new View.OnClickListener(){
  public void onClick(View v){
    /*取得服务启动后一天的 0 点 0 分 0 秒的 millsTime*/
    Calendar calendar=Calendar.getInstance();
    calendar.add(Calendar.DATE,1);
    calendar.set(Calendar.HOUR_OF_DAY,0);
    calendar.set(Calendar.MINUTE,0);
```

```
            calendar.set(Calendar.SECOND,0);
            calendar.set(Calendar.MILLISECOND,0);
            long startTime=calendar.getTimeInMillis();
            /*重复运行的间隔时间*/
            long repeatTime=24*60*60*1000;
            /*将更换桌面背景的排程添加到 AlarmManager 中*/
            Intent intent = new Intent(example11.this,MyReceiver.class);
            PendingIntent sender = PendingIntent.getBroadcast(
                                      example11.this, 0, intent, 0);
            AlarmManager am = (AlarmManager)getSystemService(
              ALARM_SERVICE);
            /*setRepeating()可让排程重复运行
                startTime 为开始运行时间
                repeatTime 为重复运行间隔
                AlarmManager.RTC 可使服务休眠时仍然会运行*/
            am.setRepeating(AlarmManager.RTC,startTime,repeatTime,
                                sender);
            /*使用 Toast 提示已启动*/
            Toast.makeText(example11.this,"服务已启动",Toast.LENGTH_SHORT)
              .show();
            /*启动后马上先运行一次换桌面背景的程序以更换今天的桌面背景*/
            Intent i = new Intent(example11.this,Change.class);
            startActivity(i);
          }
        });
```

上述代码的执行流程如下所示。

a．获取服务启动后一天的 0 点 0 分 0 秒的时间 millsTime。

b．重复运行的间隔时间。

c．将更换桌布的排程添加到 AlarmManager 中。

d．通过 setRepeating()让排程处理重复运行。

e．使用 Toast 提示已经启动。

f．启动后马上先运行一次换桌面背景的程序以更换今天的桌面背景。

⑦ 定义 onClick(View v)来监听单击终止服务按钮的事件，具体代码如下所示。

```
        /*设置终止服务的 Button*/
        mButton2=(Button) findViewById(R.id.myButton2);
        mButton2.setOnClickListener(new View.OnClickListener()
        {
          public void onClick(View v)
          {
            Intent intent = new Intent(example.this,MyReceiver.class);
            PendingIntent sender = PendingIntent.getBroadcast(
                                      example11.this, 0, intent, 0);
            /*从 AlarmManager 中删除调度*/
            AlarmManager am = (AlarmManager)getSystemService(
              ALARM_SERVICE);
            am.cancel(sender);
            /*使用 Toast 提示已终止*/
            Toast.makeText(example11.this,"服务已终止",Toast.LENGTH_SHORT)
              .show();
```

```
        }
      });
    }
```

⑧ 定义方法 initSettingData()从数据库中获取设置的值，具体代码如下所示。

```
/*由数据库中取得设置值的方法*/
private void initSettingData()
{
  map=new LinkedHashMap<Integer,Integer>();
  db=new Dai(example11.this);
  Cursor cur=db.select();
  while(cur.moveToNext()){
    map.put(cur.getInt(0),cur.getInt(1));
  }
  cur.close();
  db.close();
}
```

⑨ 设置单击按钮后跳出的选择图片的 Dialog 对话框，然后设置预览画面的文件名与 ImageView 图像，最后改变画面显示的设置图文件的文件名，并将更改的设置存入数据库。具体代码如下所示。

```
/*设置 Button 的 OnClickListener 的方法*/
private Button initButton(Button b,final TextView t,final int id)
{
  b.setOnClickListener(new View.OnClickListener()
  {
    public void onClick(View v)
    {
      /*设置单击 Button 后跳出的选择图片的 Dialog*/
      new AlertDialog.Builder(example11.this)
        .setIcon(R.drawable.pic_icon)
        .setTitle("请选择图片！ ")
        .setSingleChoiceItems(bgName,map.get(id),
         new DialogInterface.OnClickListener()
         {
           public void onClick(DialogInterface dialog,int which)
           {
             tmpWhich=which;
             /*选择图片后跳出预览图文件的窗口*/
             View view=inflater.inflate(R.layout.preview, null);
             TextView message=(TextView) view.findViewById(
               R.id.bgName);
             /*设置预览画面的文件名与 ImageView*/
             message.setText(bgName[which]);
             ImageView mView01 = (ImageView)view.findViewById(
               R.id.bgImage);
             mView01.setImageResource(bg[which]);
             Toast toast=Toast.makeText(example11.this,"", Toast.LENGTH_SHORT);
             toast.setView(view);
             toast.show();
           }
         })
        .setPositiveButton("确定",
```

```
                new DialogInterface.OnClickListener()
              {
                public void onClick(DialogInterface dialog,int which1)
                {
                  /*改变画面显示的设置图文件文件名*/
                  t.setText(bgName[tmpWhich]);
                  /*改变 map 中的值*/
                  map.put(id,tmpWhich);
                  /*将更改的设置保存到数据库*/
                  saveData(id,tmpWhich);
                }
              })
              .setNegativeButton("取消",new DialogInterface.OnClickListener()
              {
                public void onClick(DialogInterface dialog,int which)
                {
                }
              }).show();
          }
        });
        return b;
    }
```

⑩ 定义方法 saveData()将设置值存储到 DB，具体代码如下所示。

```
    private void saveData(int id,int value)
    {
      db=new Dai(example11.this);
      db.update(id,value);
      db.close();
    }
}
```

（2）编写文件 MyReceiver.java，运行后更换桌面背景，具体代码如下所示。

```
/*运行更换桌面背景的 Receiver*/
public class Receiver extends BroadcastReceiver
{
  @Override
  public void onReceive(Context context, Intent intent)
  {
    /*create Intent，调用 Change.class*/
    Intent i = new Intent(context, Change.class);
    Bundle bundleRet = new Bundle();
    bundleRet.putString("STR_CALLER", "");
    i.putExtras(bundleRet);
    i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(i);
  }
}
```

（3）编写文件 Change.java，在运行时更换桌面背景的 Activity，并声明存放图文件 id 的数组 bg，具体代码如下所示。

```
/*实际运行更换桌面背景的 Activity*/
public class Change extends Activity
{
```

```
    /*声明存放图文件 id 的数组 bg*/
    private static final int[ ] bg =
      {R.drawable.b01,R.drawable.b02,R.drawable.b03,R.drawable.b04,
      R.drawable.b05,R.drawable.b06,R.drawable.b07};
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      /*载入 progress.xml Layout*/
      setContentView(R.layout.pro);
      /*获取今天是星期几*/
      Calendar ca=Calendar.getInstance();
      int dayOfWeek=ca.get(Calendar.DAY_OF_WEEK)-1;

      /*从数据库中获取今天应该换哪一张背景*/
      int DailyBg=0;
      String selection = "DailyId=?";
      String[ ] selectionArgs = new String[ ]{""+dayOfWeek};
      Dai db=new Dai(Change.this);
      Cursor cur=db.select(selection,selectionArgs);
      while(cur.moveToNext())
      {
        DailyBg=cur.getInt(0);
      }
      cur.close();
      db.close();
      /*如果 DailyBg==99 则代表没设置，所以不运行*/
      if(DailyBg!=99)
      {
        Bitmap bmp=BitmapFactory.decodeResource
        (getResources(), bg[DailyBg]);
        try
        {
          super.setWallpaper(bmp);
        }
        catch (IOException e)
        {
          e.printStackTrace();
        }
      }
      finish();
    }
}
```

通过上述代码获取今天是星期几，然后从数据库中取得今天应该换哪一张背景，如果 DailyBg==99，则代表没设置，不运行。

（4）编写文件 Dai.java，其具体实现流程如下所示。

① 声明变量并定义构造器 Dai(Context context)，具体代码如下所示。

```
public class Dai extends SQLiteOpenHelper
{
    /*变量声明*/
```

```
private final static String DATABASE_NAME = "dailyBG_db";
private final static int DATABASE_VERSION = 1;
private final static String TABLE_NAME = "dailySetting_table";
public final static String FIELD1 = "DailyId";
public final static String FIELD2 = "DailyBg";
public SQLiteDatabase sdb;

/*构造器*/
public Dai(Context context)
{
  super(context, DATABASE_NAME, null, DATABASE_VERSION);
  sdb= this.getWritableDatabase();
}
```

② 如果 Table 不存在，则建 table 表格，并存入初始的数据到数据库对象 DB，具体代码如下所示。

```
public void onCreate(SQLiteDatabase db)
{
  /*Table 不存在就创建 table*/
  String sql = "CREATE TABLE IF NOT EXISTS "+TABLE_NAME+"("+FIELD1
      +" INTEGER primary key, "+FIELD2+" INTEGER)";
  db.execSQL(sql);

  /*存入初始的数据到 DB*/
  sdb=db;
  insert(0,99);
  insert(1,99);
  insert(2,99);
  insert(3,99);
  insert(4,99);
  insert(5,99);
  insert(6,99);
}
@Override
public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion)
{
}
public Cursor select()
{
  Cursor cursor=sdb.query(TABLE_NAME,null,null,null,null,null,null);
  return cursor;
}
```

③ 定义方法 select()来检索数据，当 select 有 where 条件时才使用此方法，具体代码如下所示。

```
public Cursor select(String selection,String[ ] selectionArgs)
{
  String[ ] columns = new String[ ] { FIELD2 };
  Cursor cursor=sdb.query(TABLE_NAME,columns,selection,
                          selectionArgs,null,null,null);
  return cursor;
}
```

④ 定义方法 insert()用于将添加的值放入 ContentValues，具体代码如下所示。

```
public long insert(int value1,int value2)
{
    /*将添加的值放入 ContentValues*/
    ContentValues cv = new ContentValues();
    cv.put(FIELD1, value1);
    cv.put(FIELD2, value2);
    long row = sdb.insert(TABLE_NAME, null, cv);
    return row;
}
```

⑤ 定义方法 delete(int id)来删除设置，具体代码如下所示。

```
public void delete(int id)
{
    String where = FIELD1 + " = ?";
    String[ ] whereValue ={ Integer.toString(id) };
    sdb.delete(TABLE_NAME, where, whereValue);
}
```

⑥ 定义方法 update()来修改设置，具体代码如下所示。

```
public void update(int id, int value)
{
    String where = FIELD1 + " = ?";
    String[ ] whereValue ={ Integer.toString(id) };
    /*将修改的值放入 ContentValues*/
    ContentValues cv = new ContentValues();
    cv.put(FIELD2, value);
    sdb.update(TABLE_NAME, cv, where, whereValue);
}
}
```

（5）编写文件 AndroidManifest.xml，在其中添加 MyReceiver 的 receiver 设置，并添加使用背景图像权限 android.permission.SET_WALLPAPER，具体代码如下所示。

```
<receiver android:name=".MyReceiver" android:process=":remote"/>
<!-- 设定 SET_WALLPAPER 权限 -->
<uses-permission android:name="android.permission.SET_WALLPAPER" />
```

执行后的效果如图 5-18 所示，选择一天并单击后面的“设置”按钮，在弹出的对话框界面中可以设置这天的背景图片，如图 5-19 所示。



图 5-18　执行效果

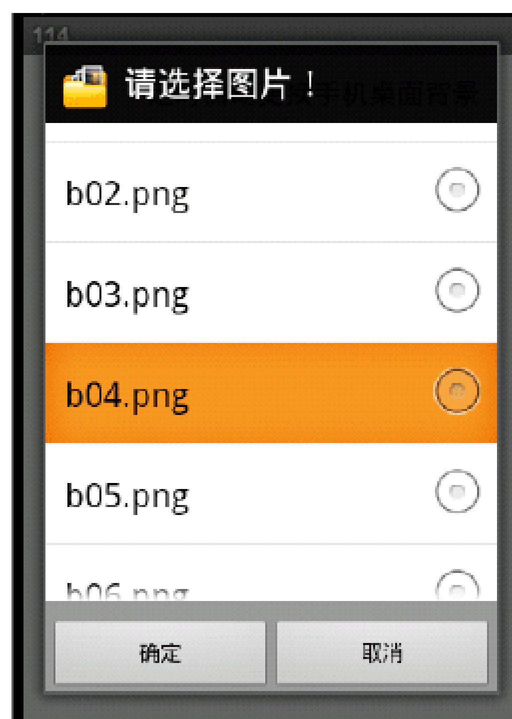


图 5-19　设置背景图片

5.11 自动显示一个开机界面

实例 081	自动显示一个开机界面
源码路径	光盘:\daima\081
视频路径	光盘:\视频\081
实例必备	081.开机发送 BOOT_COMPLETED 广播信息.pdf

5.11.1 实例说明

打开手机设备后一般都会自动显示一个特定的开机界面。在本实例中，包含了一个主程序 Activity 和一个 Broadcast Receiver 类，只要运行此程序一次，以后一开机就会运行这个程序。

5.11.2 具体实现

（1）编写文件 example.java，在此文件中定义主界面 Activity，设置程序只要运行一次，就会在以后开机时自动运行，在运行时使用 TextView 文字在屏幕中输出提示，具体代码如下所示。

```
  /*本程序只要运行一次，就会在日后开机时自动运行*/
  private TextView mTextView01;

  /** Called when the activity is first created.*/
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /*为了快速示意，程序仅以欢迎的 TextView 文字作为展示*/
    mTextView01 = (TextView)findViewById(R.id.myTextView1);
    mTextView01.setText(R.string.str_welcome);
  }
}
```

（2）编写文件 IntentReceiver.java，在此文件中定义类 IntentReceiver，在其内部覆盖了 onReceive()方法，此方法会接收来自系统的广播。方法 onReceive()的功能是唤醒自己，所以在传入 Intent 的参数中的第二个参数是指定 Activity 的 class，最后以 startActivity()方法打开并运行程序，具体代码如下所示。

```
/*捕捉 android.intent.action.BOOT_COMPLETED 的 Receiver 类*/
public class IntentReceiver extends BroadcastReceiver
{
  @Override
  public void onReceive(Context context, Intent intent)
  {
    /*当收到 Receiver 时，指定打开此程序（EX06_16.class）*/
    Intent mBootIntent = new Intent(context, example115.class);
```

```
        /*设置 Intent 打开为 FLAG_ACTIVITY_NEW_TASK*/
        mBootIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        /*将 Intent 以 startActivity 传送给操作系统*/
        context.startActivity(mBootIntent);
    }
}
```

执行后将会显示设置的开机欢迎语，如图 5-20 所示。

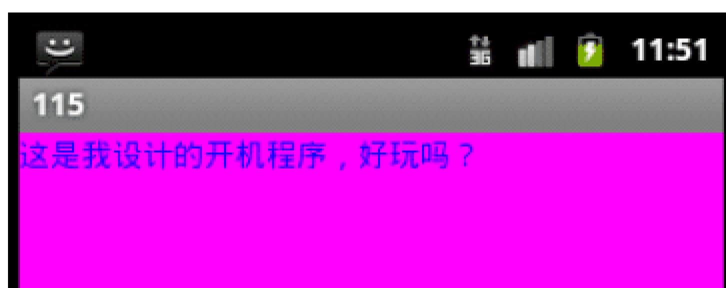


图 5-20　执行效果

5.12　自动控制系统服务

实例 082	自动控制系统服务
源码路径	光盘:\daima\082
视频路径	光盘:\视频\082
实例必备	082.Service 服务的作用.pdf

5.12.1　实例说明

Android 系统的 Service 对象是以分时进程的方式运行的，这表示即便是通过 Activity 启动 Service，也不会在相同的 process 进程中运行，而是各自属于不同的程序。在本实例中，通过 Activity 分别实现开始和终止 Service 服务，但假如直接编写 Service，也无法证明服务是否真的在“后台运行”，所以在实例中演示了如何在 Service 中开启一个 Runnable 进程的方法，每一秒都在 console 中输出运行的秒数，这样可以证明系统服务真的处于“正在运行”中。

5.12.2　具体实现

1．编写主程序文件

在主程序中设置了两个按钮事件，一个负责打开 Service，另一个负责关闭已打开的 Service。打开服务的方法与打开 Activity 的方法不同，虽然都是通过 Intent 来达成，但是前者是 startService()，后者是 startActivity()，而关闭 Service 的方法是 stopService()。无论是打开还是关闭服务，在构造的 Intent 对象中第一个传入的参数都是这个 Activity (Package)的 Context，第二个参数是服务的类*.class，主程序文件的主要代码如下所示。

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mButton01 = (Button)findViewById(R.id.myButton1);
    /*开始启动系统服务按钮事件*/
```

```
        mButton01.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v)
          {
            /*构建 Intent 对象，指定打开对象为 mService1 服务*/
            Intent i = new Intent( example.this, Service1.class );
            /*设置新 TASK 的方式*/
            i.setFlags( Intent.FLAG_ACTIVITY_NEW_TASK );
            /*以 startService 方法启动 Intent*/
            startService(i);
          }
        });
        mButton02 = (Button)findViewById(R.id.myButton2);
        /*关闭系统服务按钮事件*/
        mButton02.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v)
          {
            /*构建 Intent 对象，指定欲关闭的对象为 mService1 服务*/
            Intent i = new Intent( example.this, Service1.class );
           /*以 stopService()方法关闭 Intent*/
            stopService(i);
          }
        });
    }
}
```

2．编写文件 Servicel.java

在同一个 Activity 的 Package 层级下新建一个名为 Service1 的自定义类，此处的类名称 Servicel 是后台运行的服务名称，此名称必须与在文件 Manifest.xml 中定义的 Service 名称相符。文件 Servicel.java 的主要代码如下所示。

```
public class Service1 extends Service
{
  private Handler objHandler = new Handler();
  private int intCounter=0;
  private Runnable mTasks = new Runnable()
  {
    public void run()
    {
      intCounter++;
      Log.i("HIPPO", "Counter:"+Integer.toString(intCounter));
      objHandler.postDelayed(mTasks, 1000);
    }
  };

  @Override
  public void onStart(Intent intent, int startId)
  {
    objHandler.postDelayed(mTasks, 1000);
```

```
        super.onStart(intent, startId);
    }
    @Override
    public void onCreate()
    {
        super.onCreate();
    }

    @Override
    public IBinder onBind(Intent intent)
    {
        return null;
    }
    @Override
    public void onDestroy()
    {
        objHandler.removeCallbacks(mTasks);
        super.onDestroy();
    }
}
```

3．编写文件 AndroidManifest.xml

在其中定义服务名称、访问权限，在此需注意<service></service>所写的位置是在</activity>之后，在</application>之前。文件 AndroidManifest.xml 的主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="irdc.example118"
    android:versionCode="1"
    android:versionName="1.0.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity
            android:name="irdc.example118.example118"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service  android:name="irdc.example118.Service1"  android:exported="true"  android:process=":remote">
</service>
    </application>
</manifest>
```

执行后的效果如图 5-21 所示，单击屏幕中的按钮后可以分别启动或关闭 Service 服务。

启动Service

停止Service

图 5-21　执行效果

第 6 章　文件操作和数据存储实战

在本书前面的实例中，曾经提到过“数据存储”和“文件操作”。数据存储是手机领域中最常见的应用之一，通过数据存储能够在移动设备中显示不同的信息。文件操作是指创建、修改或删除当前手机设备中的文件。本章将通过具体实例的实现流程详细讲解在 Android 系统中实现文件操作和数据存储的方法。

6.1　修改/删除手机中的文件

实例 083	修改/删除手机中的文件
源码路径	光盘:\daima\083
视频路径	光盘:\视频\083
实例必备	083.SharedPreferences 存储.pdf

6.1.1　实例说明

在手机系统中，除了能够查看其中的文件和文件夹外，还经常需要操作管理一些文件。本实例先实现文件浏览功能，然后再添加修改/删除文件功能，实现对指定文件或文件夹名字的修改或删除操作。这些名字修改和文件删除等操作都是通过 Java I/O 实现的。

6.1.2　具体实现

（1）编写主程序文件，其具体实现流程如下所示。

① 通过如下代码声明需要的 3 个对象。

☑ items：存放显示的名称。

☑ paths：存放文件路径。

☑ rootPath：起始目录。

```
/*对象声明
    items：存放显示的名称
    paths：存放文件路径
    rootPath：起始目录
*/
private List<String> items=null;
private List<String> paths=null;
private String rootPath="/";
private TextView mPath;
```

```
private View myView;
private EditText myEditText;

@Override
protected void onCreate(Bundle icicle)
{
  super.onCreate(icicle);
  /*载入 main.xml Layout*/

  setContentView(R.layout.main);
  /*初始化 mPath，用以显示目前路径*/
  mPath=(TextView)findViewById(R.id.mPath);
  getFileDir(rootPath);
}
```

② 定义方法 getFileDir(String filePath)，用于获取手机内的文件架构。

```
private void getFileDir(String filePath)
{
  /*设置目前所在路径*/
  mPath.setText(filePath);
  items=new ArrayList<String>();
  paths=new ArrayList<String>();

  File f=new File(filePath);
  File[ ] files=f.listFiles();

  if(!filePath.equals(rootPath))
  {
    /*第一笔设置为“回到根目录”*/
    items.add("b1");
    paths.add(rootPath);
    /*第二笔设置为“回到上一层”*/
    items.add("b2");
    paths.add(f.getParent());
  }
  /*将所有文件添加到 ArrayList 中*/
  for(int i=0;i<files.length;i++)
  {
    File file=files[i];
    items.add(file.getName());
    paths.add(file.getPath());
  }
  /*使用自定义的 MyAdapter 将数据传入 ListActivity*/
  setListAdapter(new MyAdapter(this,items,paths));
}
```

上述方法的实现流程如下所示。

a．设置目前所在路径。

b．分别实现“回到根目录”和“回到上一层”操作。

c．使用自定义的 MyAdapter 将数据传入 ListActivity。

③ 设置 ListItem 被单击时要做的动作处理事件 onListItemClick(ListView l,View v,int position,long id)。

如果是文件夹，则运行方法 getFileDir()，如果是文件，则运行方法 fileHandle()，具体代码如下所示。

```
@Override
protected void onListItemClick(ListView l,View v,int position,long id)
{
  File file = new File(paths.get(position));
  if(file.isDirectory())
  {
    /*如果是文件夹，则运行 getFileDir()*/
    getFileDir(paths.get(position));
  }
  else
  {
    /*如果是文件，则调用 fileHandle()*/
    fileHandle(file);
  }
}
```

④ 定义方法 fileHandle(final File file)，用于处理文件。如果 which==0，则选择要打开的文件；如果 which==1，则修改文件名；如果 which 是其他值，则删除选中的文件，具体实现代码如下所示。

```
/*处理文件的方法*/
private void fileHandle(final File file){
  /*单击文件时的 OnClickListener*/
  OnClickListener listener1=new DialogInterface.OnClickListener()
  {
    public void onClick(DialogInterface dialog,int which)
    {
      if(which==0)
      {
        /*选择的 item 为打开文件*/
        openFile(file);
      }
      else if(which==1)
      {
        /*更改选择的 item 文件名*/
        LayoutInflater factory=LayoutInflater.from(example.this);
        /*初始化 myChoiceView，使用 rename_alert_dialog 为 layout*/
        myView=factory.inflate(R.layout.rename_alert_dialog,null);
        myEditText=(EditText)myView.findViewById(R.id.mEdit);
        /*将原始文件名先放入 EditText 中*/
        myEditText.setText(file.getName());
        /*新建一个更改文件名的对话框，然后监听这个对话框中的确定按钮的*/
        OnClickListener listener2=
        new DialogInterface.OnClickListener()
        {
          public void onClick(DialogInterface dialog, int which)
          {
            /*获取修改后的文件路径*/
            String modName=myEditText.getText().toString();
            final String pFile=file.getParentFile().getPath()+" / ";
            final String newPath=pFile+modName;
            /*判断文件名是否已存在*/
```

```
        if(new File(newPath).exists())
        {
          /*排除修改文件名时没修改直接送出的状况*/
          if(!modName.equals(file.getName()))
          {
            /*跳出 Alert 警告文件名重复，并确认是否修改*/
            new AlertDialog.Builder(example12.this)
                .setTitle("注意!")
                .setMessage("已经存在，是否要覆盖?")
                .setPositiveButton("确定",
                 new DialogInterface.OnClickListener()
                {
                  public void onClick(DialogInterface dialog,
                                        int which)
                  {
                    /*文件名重复仍然修改会覆盖已存在的文件*/
                    file.renameTo(new File(newPath));
                    /*重新产生文件列表的 ListView*/
                    getFileDir(pFile);
                  }
                })
                .setNegativeButton("取消",
                 new DialogInterface.OnClickListener()
                {
                  public void onClick(DialogInterface dialog,
                                        int which)
                  {
                  }
                }).show();
          }
        }
        else
        {
          /*文件名不存在，直接做修改动作*/
          file.renameTo(new File(newPath));
          /*重新产生文件列表的 ListView*/
          getFileDir(pFile);
        }
      }
    };
    /*create 更改文件名时跳出的 Dialog*/
    AlertDialog renameDialog=
      new AlertDialog.Builder(example.this).create();
    renameDialog.setView(myView);
    /*设置更改文件名单击确认后的 Listener*/
    renameDialog.setButton("确定",listener2);
    renameDialog.setButton2("取消",
    new DialogInterface.OnClickListener()
    {
      public void onClick(DialogInterface dialog, int which)
      {
```

```
                }
            });
            renameDialog.show();
        }
        else
        {
            /*选择的 item 为删除文件*/
            new AlertDialog.Builder(example.this).setTitle("请注意!")
                    .setMessage("确定删除?")
                    .setPositiveButton("确定",
                     new DialogInterface.OnClickListener()
                    {
                       public void onClick(DialogInterface dialog,
                                                    int which)
                       {
                          /*删除文件*/
                          file.delete();
                          getFileDir(file.getParent());
                       }
                    })
                    .setNegativeButton("取消",
                     new DialogInterface.OnClickListener()
                    {
                       public void onClick(DialogInterface dialog,
                                                    int which)
                       {
                       }
                    }).show();
        }
    }
};
```

⑤ 当用户选择一个文件时系统会自动弹出一个要如何处理文件的 ListDialog 对话框，对应代码如下所示。

```
    String[ ] menu={"打开","更名","删除"};
    new AlertDialog.Builder(example12.this)
        .setTitle("准备干啥?")
        .setItems(menu,listener1)
        .setPositiveButton("取消",
         new DialogInterface.OnClickListener()
        {
           public void onClick(DialogInterface dialog, int which)
           {
           }
        })
        .show();
}
```

⑥ 定义方法 openFile(File f)用于在手机上打开指定的文件，具体代码如下所示。

```
/*在手机上打开文件的方法*/
private void openFile(File f)
{
```

```
        Intent intent = new Intent();
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.setAction(android.content.Intent.ACTION_VIEW);
        /*调用 getMIMEType()来取得 MimeType*/
        String type = getMIMEType(f);
        /*设置 intent 的 file 与 MimeType*/
        intent.setDataAndType(Uri.fromFile(f),type);
        startActivity(intent);
    }
```

⑦ 定义方法 getMIMEType(File f)用于判断指定文件的类型，具体代码如下所示。

```
    /*判断文件 MimeType 的方法*/
    private String getMIMEType(File f)
    {
        String type="";
        String fName=f.getName();
        /*取得扩展名*/
        String end=fName.substring(fName.lastIndexOf(".")+1,
                                    fName.length()).toLowerCase();
        /*依附文件名的类型决定 MimeType*/
        if(end.equals("m4a")||end.equals("mp3")||end.equals("mid")
            ||end.equals("xmf")||end.equals("ogg")||end.equals("wav"))
        {
            type = "audio";
        }
        else if(end.equals("3gp")||end.equals("mp4"))
        {
            type = "video";
        }
        else if(end.equals("jpg")||end.equals("gif")||end.equals("png")
                    ||end.equals("jpeg")||end.equals("bmp"))
        {
            type = "image";
        }
        else
        {
            /*如果无法直接打开，则弹出软件列表供用户选择*/
            type="*";
        }
        type += "/*";
        return type;
    }
}
```

（2）编写文件 MyAdapter.java，此文件的具体实现流程如下所示。

① 通过如下代码分别声明 4 个变量。

```
    /*变量声明*/
    private LayoutInflater mInflater;
    private Bitmap mIcon1;
    private Bitmap mIcon2;
    private Bitmap mIcon3;
    private Bitmap mIcon4;
```

```
private List<String> items;
private List<String> paths;
```

对上述 4 个变量的具体说明如下所示。

☑ mIcon1：回到根目录的图文件。

☑ mIcon2：回到上一层的图档。

☑ mIcon3：文件夹的图文件。

☑ mIcon4：文件的图档。

② 定义构造器 MyAdapter，分别传入参数 items、paths 和 mInflater，最后赋值前面定义的 4 个变量，具体代码如下所示。

```
/*MyAdapter 的构造器，传入 3 个参数 */
public MyAdapter(Context context,List<String> it,List<String> pa)
{
  /*参数初始化*/
  mInflater = LayoutInflater.from(context);
  items = it;
  paths = pa;
  mIcon1 = BitmapFactory.decodeResource(context.getResources(),
                                       R.drawable.back01);
  mIcon2 = BitmapFactory.decodeResource(context.getResources(),
                                       R.drawable.back02);
  mIcon3 = BitmapFactory.decodeResource(context.getResources(),
                                       R.drawable.folder);
  mIcon4 = BitmapFactory.decodeResource(context.getResources(),
                                       R.drawable.doc);
}
```

③ 使用自定义的 file_row 作为 Layout 布局样式，然后分别设置“回到根目录”的文字与 icon 和“回到上一层”的文字与 icon，具体代码如下所示。

```
@Override
public View getView(int position,View convertView,ViewGroup parent)
{
  ViewHolder holder;

  if(convertView == null)
  {
    /*使用自定义的 file_row 作为 Layout*/
    convertView = mInflater.inflate(R.layout.file_row, null);
    /*初始化 holder 的 text 与 icon*/
    holder = new ViewHolder();
    holder.text = (TextView) convertView.findViewById(R.id.text);
    holder.icon = (ImageView) convertView.findViewById(R.id.icon);

    convertView.setTag(holder);
  }
  else
  {
    holder = (ViewHolder) convertView.getTag();
  }
  File f=new File(paths.get(position).toString());
```

```
        /*设置“回到根目录”的文字与 icon*/
        if(items.get(position).toString().equals("b1"))
        {
          holder.text.setText("Back to / ");
          holder.icon.setImageBitmap(mIcon1);
        }
        /*设置“回到上一层”的文字与 icon*/
        else if(items.get(position).toString().equals("b2"))
        {
          holder.text.setText("Back to ...");
          holder.icon.setImageBitmap(mIcon2);
        }
        /*设置“文件或文件夹”的文字与 icon*/
        else
        {
          holder.text.setText(f.getName());
          if(f.isDirectory())
          {
            holder.icon.setImageBitmap(mIcon3);
          }
          else
          {
            holder.icon.setImageBitmap(mIcon4);
          }
        }
        return convertView;
      }
      /*class ViewHolder*/
      private class ViewHolder
      {
        TextView text;
        ImageView icon;
      }
    }
```

执行后的效果如图 6-1 所示。当选择一个文件夹时会弹出一个操作对话框，在对话框中可以选择“打开文件”“改名”“删除”这 3 种操作，如图 6-2 所示。

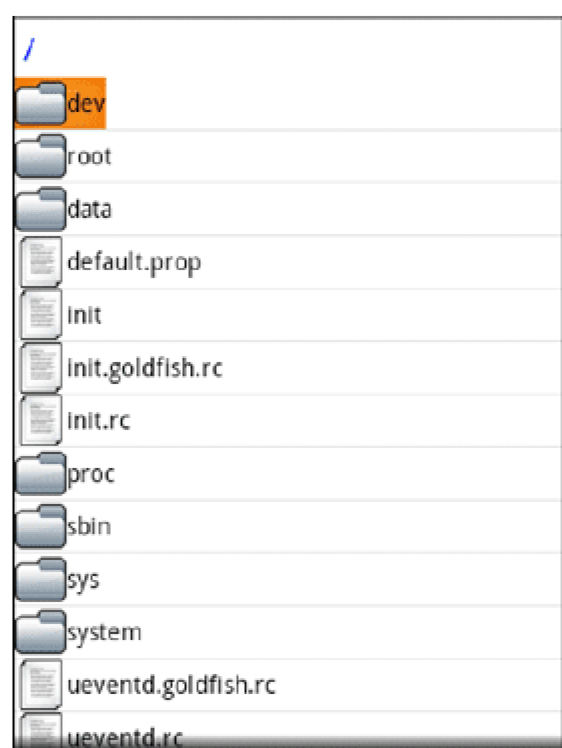


图 6-1　执行效果



图 6-2　文件信息

6.2 显示在 SharedPreferences 中存储的信息

实例 084	在屏幕中显示 SharedPreferences 存储的信息
源码路径	光盘:\daima\084
视频路径	光盘:\视频\084
实例必备	084.文件存储.pdf

6.2.1 实例说明

SharedPreferences 是 Android 提供的用来存储一些简单配置信息的机制。例如，一些默认欢迎语、登录的用户名和密码等。SharedPreferences 以键值对的方式存储，这样开发人员可以很方便地实现读取和存入。

SharedPreferences 类似于 Windows 系统上的 ini 配置文件，但是可以分为多种权限，可以全局共享访问。虽然此存储方式的整体效率不是特别高，但是对于常规的轻量级而言，比 SQLite 好很多，如果存储量不大，可以考虑自定义文件格式。XML 处理时 Dalvik 会通过自带底层的本地 XML Parser 解析，例如 XMLpull 方式，这样对于内存资源占用比较好。

6.2.2 具体实现

编写文件 exampleHelper.java，主要代码如下所示。

```
public class exampleHelper {
	SharedPreferences sp;
	SharedPreferences.Editor editor;

	Context context;

	public exampleHelper(Context c,String name){
		context = c;
		sp = context.getSharedPreferences(name, 0);
		editor = sp.edit();
	}

	public void putValue(String key, String value){
		editor = sp.edit();
		editor.putString(key, value);
		editor.commit();
	}
	public String getValue(String key){
		return sp.getString(key, null);
	}
}
```

编写文件 exampleuse.java，主要代码如下所示。

```
public class exampleuse extends Activity {
	public final static String COLUMN_NAME ="name";
```

```
    public final static String COLUMN_MOBILE ="mobile";
    exampleHelper sp;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);

        sp = new exampleHelper(this, "contacts");

        //设置存储的信息
        sp.putValue(COLUMN_NAME, "Mr WANG");
        sp.putValue(COLUMN_MOBILE, "1506907XXXX");


        //2. to fetch the value
        String name = sp.getValue(COLUMN_NAME);
        String mobile = sp.getValue(COLUMN_MOBILE);

        TextView tv = new TextView(this);
        tv.setText("NAME:"+ name + "\n" + "MOBILE:" + mobile);
        setContentView(tv);
    }
}
```

执行后的效果如图 6-3 所示。

```
NAME:Mr WANG
MOBILE:1506907XXXX
```

图 6-3　执行效果

在上述实例代码中，因为 pack_name 为 example，所以存放数据的路径如下。

```
data/data/example/share_prefs/contacts.xml
```

文件 contacts.xml 的内容如下。

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="mobile">1506907XXXX</string>
<string name="name">Mr WANG</string>
</map>
```

6.3　添加/删除 SQLite 中的数据

实例 085	演示数据添加、删除等操作
源码路径	光盘:\daima\085
视频路径	光盘:\视频\085
实例必备	085.最常用的 SQLite.docx.pdf ① SQLite 基础 ② SQLite 数据类型 ③ SQLiteDatabase 介绍

6.3.1 实例说明

在数据库编程中，通常使用 SQL 语句来操作数据库中的数据，如添加、删除、修改等操作。在 Android 开发应用中，也可以操作存储在数据库中的数据，分别实现对这些数据的添加、删除和修改操作。在本实例中，使用 SQLite 存储方式来保存数据，并通过编程方式操作其中的数据。

6.3.2 具体实现

编写主程序文件，其具体实现流程如下所示。

（1）定义继承于 SQLiteOpenHelper 的类 DatabaseHelper，具体代码如下所示。

```
private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context context) {
                super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
                String sql = "CREATE TABLE " + TABLE_NAME + " (" + TITLE
                                + " text not null, " + BODY + " text not null " + ");";
                Log.i("haiyang:createDB=", sql);
                db.execSQL(sql);
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        }
}
```

在上述代码中，类 DatabaseHelper 继承了 SQLiteOpenHelper 类，并且重写了 onCreate()和 onUpgrade()方法。在 onCreate()方法中首先构造一条 SQL 语句，然后调用 db.execSQL(sql)执行 SQL 语句。这条 SQL 语句生成了一张数据库表。

此处的 SQLiteOpenHelper 是一个辅助类，其功能是生成一个数据库，并管理数据库的版本。当在程序中调用此类中的方法 getWritableDatabase()或者 getReadableDatabase()时，如果没有数据，则 Android 系统就会自动生成一个数据库。SQLiteOpenHelper 同时也是一个抽象类，其主要功能是通过如下 3 个方法实现的。

- ☑ onCreate(SQLiteDatabase)：在数据库第一次生成时会调用这个方法，一般在该方法中生成数据库表。
- ☑ onUpgrade(SQLiteDatabase, int, int)：当数据库需要升级时，Android 系统会主动调用此方法。一般在这个方法中删除数据表，并建立新的数据表，当然是否需要做其他操作，完全取决于应用的需求。
- ☑ onOpen(SQLiteDatabase)：这是当打开数据库时的回调方法，一般不会用到。

（2）编写按钮处理事件，单击“插入两条数据”按钮后插入两条新的记录到数据库中的 diary 表中，并且在屏幕的 title（标题）区域提示操作成功，如图 6-4 所示。

单击“插入两条数据”按钮后执行监听器中的 onClick()方法，并最终执行 insertItem()方法以插入两条数据，具体代码如下所示。

```
/*插入两条数据*/
private void insertItem() {
	SQLiteDatabase db = mOpenHelper.getWritableDatabase();
	String sql1 = "insert into " + TABLE_NAME + " (" + TITLE + ", " + BODY
			+ ") values('haiyang', 'Android 很好');";
	String sql2 = "insert into " + TABLE_NAME + " (" + TITLE + ", " + BODY
			+ ") values('icesky', 'Android 很好');";
	try {
		Log.i("haiyang:sql1=", sql1);
		Log.i("haiyang:sql2=", sql2);
		db.execSQL(sql1);
		db.execSQL(sql2);
		setTitle("成功插入两条数据");
	} catch (SQLException e) {
		setTitle("插入失败");
	}
}
```

在上述代码中，sql1 和 sql2 是标准的实现插入操作的 SQL 语句，Log.i()会将参数内容打印到日志中，并且打印级别是 Info 级别，db.execSQL(sql1)的功能是执行 SQL 语句。

（3）单击“查询数据”按钮后会在屏幕的 title 区域中显示当前数据表中的数据条数，因为刚才插入了两条数据，所以此时显示为两条，如图 6-5 所示。

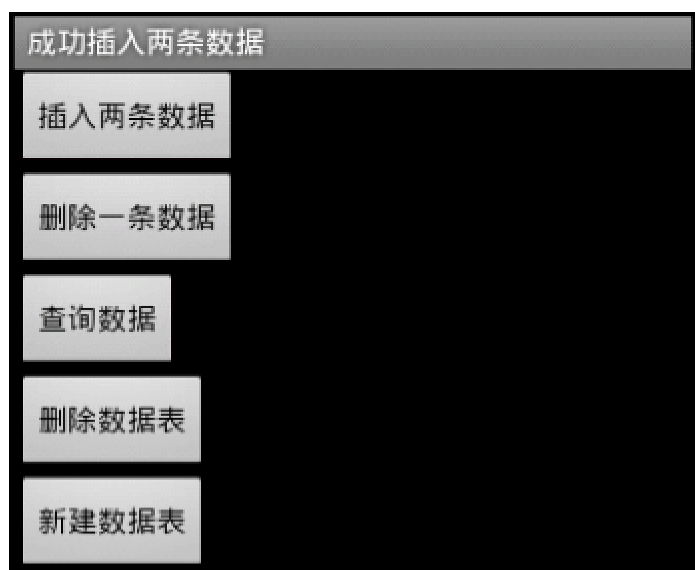


图 6-4　插入成功



图 6-5　查询数据

单击“查询数据”按钮后会执行 showItems()方法，具体代码如下所示。

```
/*在屏幕的 title 区域显示当前数据表当中的数据条数*/
private void showItems() {
	/*得到一个可写的数据库*/
	SQLiteDatabase db = mOpenHelper.getReadableDatabase();
	String col[ ] = { TITLE, BODY };
	Cursor cur = db.query(TABLE_NAME, col, null, null, null, null, null);
	/*通过 getCount()方法，可以得到 Cursor 中数据的个数*/
	Integer num = cur.getCount();
	setTitle(Integer.toString(num) + "  条记录");
}
}
```

在上述代码中，通过如下代码将查询到的数据放到一个 Cursor 中。

```
Cursor cur = db.query(TABLE_NAME, col, null, null, null, null, null)
```

通过上述语句，在 Cursor 中封装了表 TABLE_NAME 中的所有数据条列。在 query()方法中包含了 7 个参数，各参数的具体说明如下。

- ☑ 第 1 个参数：代表数据库中表的名字，如本实例中，表的名字就是 TABLE_NAME，也就是 diary。
- ☑ 第 2 个参数：代表想要返回数据包含的列的信息。本实例中想要得到的列有 title、body。将这两个列的名字放到字符串数组中。
- ☑ 第 3 个参数 selection：相当于 SQL 语句的 where 部分，如果想返回所有的数据，则直接置为 null。
- ☑ 第 4 个参数 selectionArgs：在 selection 部分可能会用到“?”操作符，此时需要用 selectionArgs 定义的字符串代替 selection 中的“?”。
- ☑ 第 5 个参数 groupBy：定义查询出来的数据是否分组，如果为 null，则说明不用分组。
- ☑ 第 6 个参数 having：相当于 SQL 语句中的 having 部分。
- ☑ 第 7 个参数 orderBy：用于描述期望的返回值是否需要排序，如果设置为 null，则说明不需要排序。

（4）单击“删除一条数据”按钮后会删除库中的一条数据，如果成功删除，在屏幕的 title 区域会显示对应的文字提示，如图 6-6 所示。

现在再单击“查询数据”按钮，看数据库中的记录少了一条。当单击“删除一条数据”按钮后会执行 deleteItem()方法，其代码如下所示。

```
/*
 * 删除其中的一条数据
 */
private void deleteItem() {
	try {
		SQLiteDatabase db = mOpenHelper.getWritableDatabase();
		db.delete(TABLE_NAME, " title = 'aa'", null);
		setTitle("删除了一条 title 为 aa 的记录");
	} catch (SQLException e) {
	}
}
```

在上述代码中，通过如下语句删除了一条 title 为 aa 的数据。如果有很多条数据 title 都为 aa，那么一并删除。在方法 delete()中各个参数的具体说明如下。

- ☑ 第 1 个参数：表示数据库表名，在此处是 TABLE_NAME，也就是 diary。
- ☑ 第 2 个参数：相当于 SQL 语句中的 where 部分，也就是描述了删除的条件。如果在第 2 个参数中有“?”，那么第 3 个参数中的字符串会依次替换在第 2 个参数当中出现的“?”。

（5）单击“删除数据表”按钮后可以删除数据表 diary，如图 6-7 所示。



图 6-6　删除一条数据



图 6-7　删除数据表

删除数据表功能是通过方法 dropTable()实现的，在此方法中构造了一个标准的删除数据表的 SQL 语句，然后执行语句 db.execSQL(sql)，具体代码如下所示。

```
 /*删除数据表*/
private void dropTable() {
	SQLiteDatabase db = mOpenHelper.getWritableDatabase();
	String sql = "drop table " + TABLE_NAME;
	try {
		db.execSQL(sql);
		setTitle("成功删除数据表：" + sql);
	} catch (SQLException e) {
		setTitle("删除错误");
	}
}}
```

（6）此时单击其他按钮，程序会出现异常，在此单击“新建数据表”按钮，如图 6-8 所示。如果此时单击“查询数据”按钮，则显示有 0 条记录，如图 6-9 所示。



图 6-8　新建表

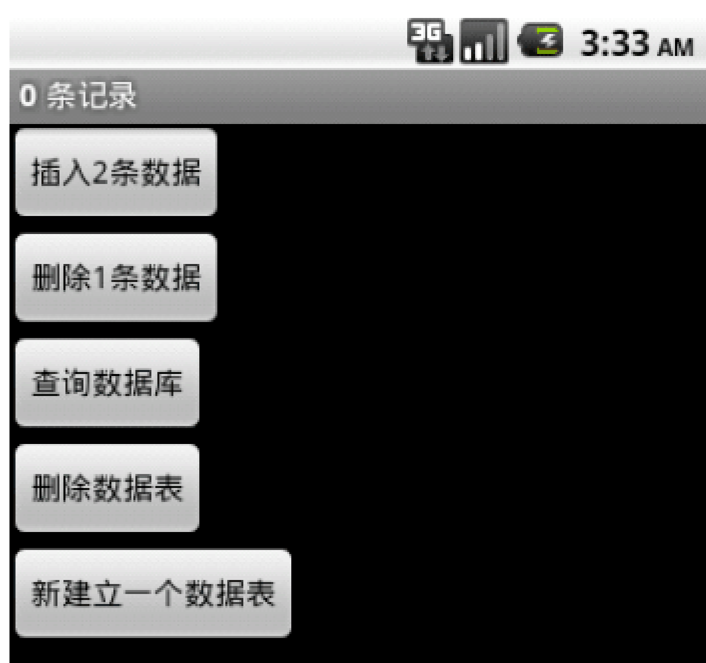


图 6-9　显示 0 条记录

创建新表功能是通过方法 CreateTable()实现的，主要代码如下所示。

```
/*重新建立数据表*/
private void CreateTable() {
	SQLiteDatabase db = mOpenHelper.getWritableDatabase();
	String sql = "CREATE TABLE " + TABLE_NAME + " (" + TITLE
			+ " text not null, " + BODY + " text not null " + ");";
	Log.i("haiyang:createDB=", sql);

	try {
		db.execSQL("DROP TABLE IF EXISTS diary");
		db.execSQL(sql);
		setTitle("重建数据表成功");
	} catch (SQLException e) {
		setTitle("重建错误");
	}
}
```

在上述代码中，如果已经存在 diary 表，则需要先删除原来的，因为在同一个数据库中不能出现两张同样名字的表。

6.4 使用 ContentProvider 存储数据

实例 086	演示 ContentProvider 的用法
源码路径	光盘:\daima\086
视频路径	光盘:\视频\086
实例必备	086.ContentProvider 存储.pdf

6.4.1 实例说明

在 Android 系统中，数据是私有的。当然，这些数据包括文件数据和数据库数据以及一些其他类型的数据。Android 中的两个程序之间可以进行数据交换，此功能是通过 ContentProvider 实现的。一个 ContentProvider 类实现了一组标准的方法接口，从而能够让其他的应用保存或读取此 ContentProvider 的各种数据类型。也就是说，一个程序可以通过实现一个 ContentProvider 的抽象接口将自己的数据暴露出来。外界根本看不到，也不用看到这个暴露的数据在应用当中是如何存储的，外界可以通过这一套标准及统一的接口与程序中的数据交互，可以读取程序的数据，也可以删除程序的数据，当然，中间也会涉及一些权限的问题。



图 6-10 出现的桌面

6.4.2 具体实现

（1）单击模拟器的MENU按钮，再单击在桌面上弹出的 Add 按钮，如图 6-10 所示。

（2）进入应用后，依次单击选择 Shortcuts、Contact 和 Create new contact，如图 6-11 所示。

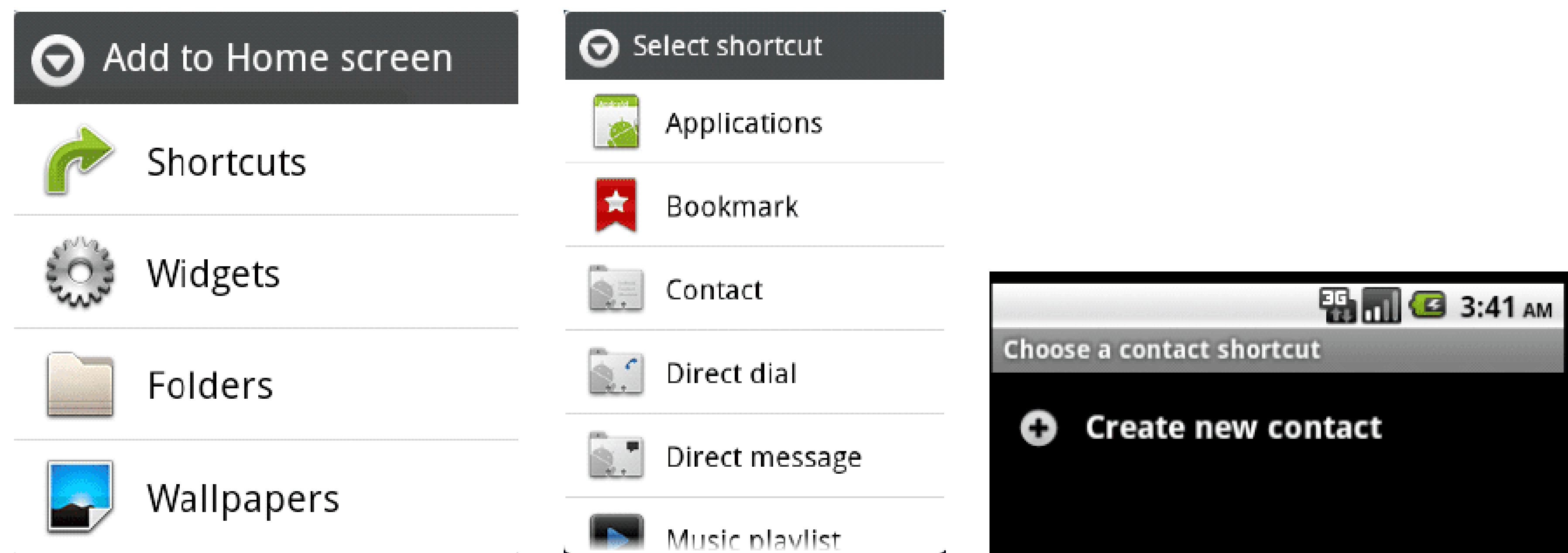


图 6-11 单击 Create new contact

（3）在弹出的界面中添加联系人姓名和电话号码信息，在这个界面中提供了添加联系人的 First Name、Last Name 和 Phone 等表单，如图 6-12 所示。

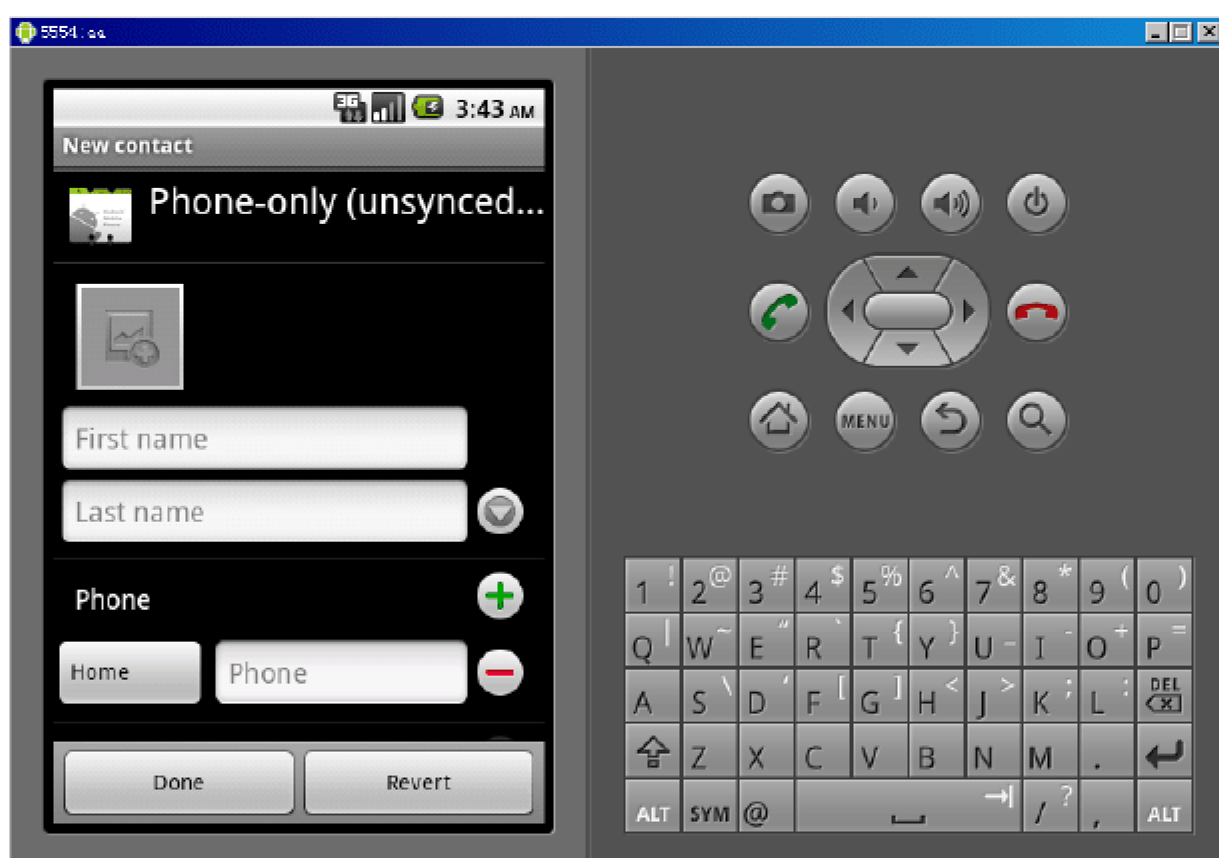


图 6-12　添加联系人姓名和电话号码

（4）填写资料完毕后，单击 Done 按钮保存，如图 6-13 所示。

（5）按照上述操作步骤，即可添加一条联系人数据，效果如图 6-14 所示。

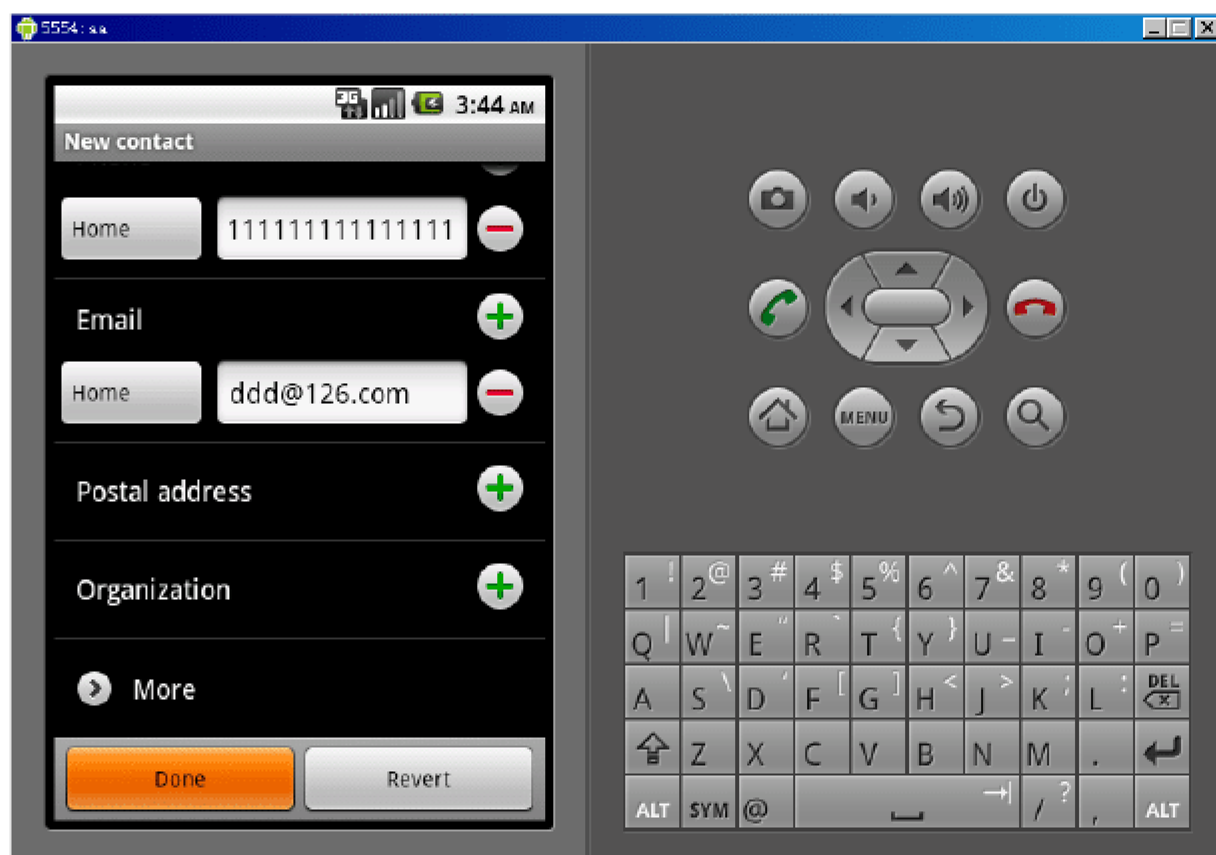


图 6-13　单击 Done 按钮保存

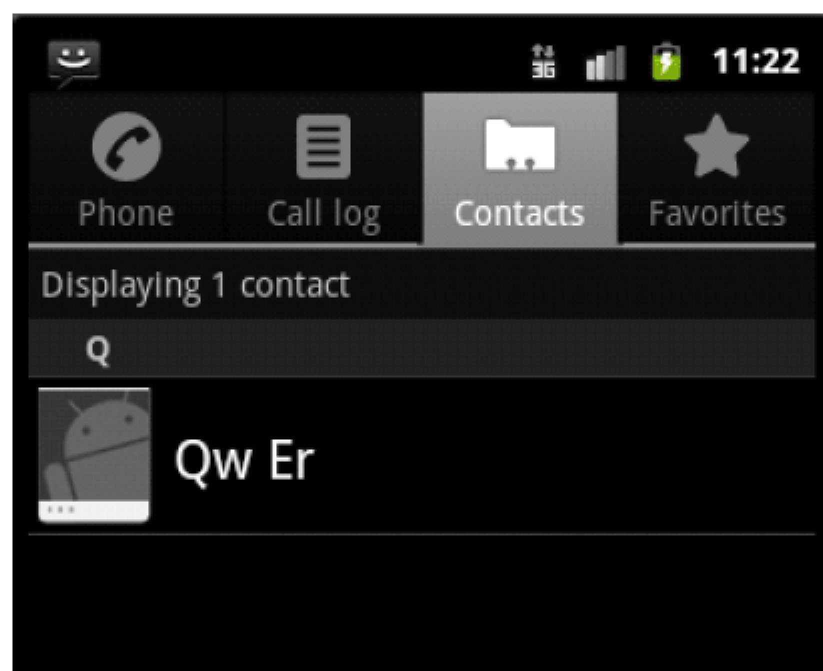


图 6-14　添加后的数据

上述存储联系人的操作过程就是将信息使用 ContentProvider 进行存储的过程。接下来再看实例文件代码，文件 example.java 的主要代码如下所示。

```
public class example extends ListActivity {
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Cursor c = getContentResolver().query(Phones.CONTENT_URI, null, null, null, null);
    startManagingCursor(c);
    ListAdapter adapter = new SimpleCursorAdapter(this,
            android.R.layout.simple_list_item_2, c,
                    new String[ ] { Phones.NAME, Phones.NUMBER },
                    new int[ ] { android.R.id.text1, android.R.id.text2 });
    setListAdapter(adapter);
  }
}
```

6.5 ContentProvider 日记本系统

实例 087	使用 ContentProvider 实现日记本功能
源码路径	光盘:\daima\087
视频路径	光盘:\视频\087
实例必备	087.网络存储.pdf ① 演练简介 ② 实现过程

6.5.1 实例说明

本书前面的内容中已经讲解过 ContentProvider 的知识，并且使用了系统中一个联系人的 ContentProvider，在本实例中，日记本功能是通过 ContentProvider 实现的，而不是直接用数据库实现。这样，外界的程序就可以访问到日记本中这个应用数据。通过本实例可以学习以下操作。

☑ 如何实现一个 ContentProvider。

☑ 理解 UriMatcher 的含义。

☑ onPrepareOptionsMenu()方法介绍。

6.5.2 具体实现

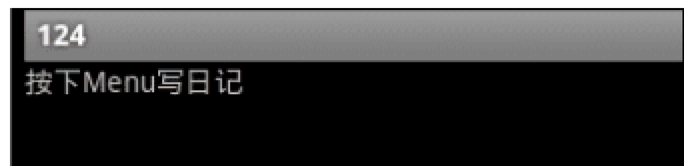


图 6-15 执行效果

运行后的效果如图 6-15 所示。

（1）本实例的主 Activity 是一个 ListActivity，对应的布局文件是 diary_list.xml，其主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ListView android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:id="@+id/android:empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="按下 Menu 写日记" />
</LinearLayout>
```

在上述代码中，ListView 的 id 必须定义成 Android:id="@+id/Android:list"的形式，这样系统才可以在 List Activity 中引用。

（2）本实例日记本的数据存储在 SQLite 数据库中，执行增、删、改、查操作时不是直接访问数据库实现的，而是通过日记本程序的 ContentProvider 实现的。接下来编写文件 Diary.java，在其中定义了 Diary 类，在此类中有一个内部静态类 DiaryColumns，和其名字意思一样，此类中主要定义了日记

本数据库的列表字段的名字，主要代码如下所示。

```
public static final class DiaryColumns implements BaseColumns {
        private DiaryColumns() { }
        public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY + "/diaries");
        public static final String CONTENT_TYPE = "vnd.Android.cursor.dir/vnd.google. diary";
        public static final String CONTENT_ITEM_TYPE = "vnd.Android.cursor.item/vnd. google.diary";
        public static final String DEFAULT_SORT_ORDER = "created DESC";
        public static final String TITLE = "title";
        public static final String BODY = "body";
        public static final String CREATED = "created";
    }
```

在上述代码中，BaseColumns 是一个接口，其中有两个变量，一个是_ID="_id"，另一个是_COUNT= "_count"。在 Android 中，每一个数据库表至少有一个字段，而且这个字段是_id。所以当构造列名的辅助类时，直接实现 BaseColumns，这样便默认地拥有了_id 字段。

在本实例的数据表中一共有 4 个字段，分别是 id、title、body 和 created。

（3）编写文件 DiaryContentProvider.java，先分析类 DiaryContentProvider，此类继承自 ContentProvider，在其中实现了 ContentProvider 的一些接口方法，并且成为日记本的一个 ContentProvider。下面通过学习这个类来详细了解实现一个自定义的 ContentProvider 的具体过程。

① 先定义一个 public staticfinal 的 Uri 类型的变量，并且命名为 CONTENT_URI。DiaryContentProvider 所能处理的 Uri 都是基于 CONTENT_URI 来构建的。需要注意的是，这个 CONTENT_URI 中的内容以 content://开头，并且全部小写，且全局唯一。

下面是本实例中一个普通的 Uri，通过分析这个 Uri，可以了解 content Uri 的构成，具体格式如下所示。

```
content://com.example124.diarycontentprovider/diaries/1
```

☑ 第 1 部分：是 content://，这部分是固定存在的，不需要做任何修改。

☑ 第 2 部分：是授权（AUTHORITY）部分，即 com.example124.diarycontentprovider，授权部分是唯一的，在程序中一般是实现的 ContentProvider 的全称，并且全都小写。这部分是与 AndroidManifest.xml 文件中的如下部分对应的。

```
<provider Android:name ="DiaryContentProvider
Android:authorities="shiyongcontentprovider. diarycontentprovider" />
```

☑ 第 3 部分：是请求数据的类型，例如，在本实例中定义的类型是 diaries。当然，这一部分可以是由 0 个片段或者多个片段构成，如果 ContentProvider 只是暴露出了一种类型的数据，那么这部分可以为空，但是如果暴露出了多种，尤其是包含子类时，就不能为空。例如，日记本程序中可以暴露出两种数据，一种是用户自己的 diaries/my，另一种是其他人的 diaries/others。

☑ 第 4 部分：是 1，当然这部分是允许为空的。如果为空，表示请求全部数据；如果不为空，表示请求特定 ID 的数据。

② 构建用户的数据存储系统。在本实例中，是将数据存储到数据库系统中。通常是将数据存储在数据库系统中，但是也可以将数据存储在其他地方，如文件系统等。

③ 实现抽象类 ContentProvider 的抽象方法，具体如下所示。

☑ public boolean onCreate()：当 ContentProvider 生成时调用此方法。

☑ public Cursor query(Uri uri, String[ ] projection, String selection,String[ ] selectionArgs, String sortOrder)：此方法返回一个 Cursor 对象作为查询结果集。

☑ public Uri insert(Uri uri, ContentValues initialValues)：此方法负责向数据集中插入一列，并返回这一列的 Uri。

☑ public int delete(Uri uri, String where, String[ ] whereArgs)：此方法负责删除指定 Uri 的数据。

☑ public int update(Uri uri, ContentValues values, String where,String[ ] whereArgs)：此方法负责更新指定 Uri 的数据。

☑ public String getType(Uri uri)：返回所给 Uri 的 MIME 类型。

④ 在 AndroidManifest.xml 文件中增加<provider>标签，例如，本实例中的实现代码如下所示。

```
<provider Android:name="DiaryContentProvider" Android:authorities="com.example124. diarycontentprovider" />
```

其中，Android:name 是实现的 ContentProvider 的类名；Android: authorities 是 content Uri 的第 2 部分，即授权部分，实现代码是 com.shiyongcontentprovider.diarycontentprovider。

（4）继续看实现 DiaryContentProvider 类的代码，因为此实例的数据是存储在数据库中，所以需要定义 DatabaseHelper 辅助类，主要代码如下所示。

```
    private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
            Log.i("jinyan", "DATABASE_VERSION=" + DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            Log.i("jinyan", "onCreate(SQLiteDatabase db)");
            String sql = "CREATE TABLE " + DIARY_TABLE_NAME + " ("
                    + DiaryColumns._ID + " INTEGER PRIMARY KEY,"
                    + DiaryColumns.TITLE + " TEXT," + DiaryColumns.BODY
                    + " TEXT," + DiaryColumns.CREATED + " TEXT" + ");";
            //
            sql ="CREATE TABLE " + DIARY_TABLE_NAME + " ("
            + DiaryColumns._ID + " INTEGER PRIMARY KEY,"
            + DiaryColumns.TITLE + " varchar(255)," + DiaryColumns.BODY
            + " TEXT," + DiaryColumns.CREATED + " TEXT" + ");";
            Log.i("jinyan", "sql="+sql);
            db.execSQL(sql);
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            Log.i("jinyan",
                    " onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)="
                            + newVersion);
            db.execSQL("DROP TABLE IF EXISTS diary");
            onCreate(db);
        }
    }
```

在上述代码中，DatabaseHelper 是用于操作数据库的辅助类，通过此类可以生成并维护数据库。

（5）在类 DiaryContentProvider 中定义需要的变量和常量，其中的常量主要是描述数据库的信息。

```
private static final String DATABASE_NAME = "database";
private static final int DATABASE_VERSION = 1;
private static final String DIARY_TABLE_NAME = "diary";
private static HashMap<String, String> sDiariesProjectionMap;
private static final int DIARIES = 1;
```

```
private static final int DIARY_ID = 2;
private static final UriMatcher sUriMatcher;
```

编写 static 模块，具体代码如下所示。

```
sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
sUriMatcher.addURI(Diary.AUTHORITY, "diaries", DIARIES);
sUriMatcher.addURI(Diary.AUTHORITY, "diaries/#", DIARY_ID);
```

上面代码中的 UriMatcher 是匹配 Uri 的一个辅助类，具体说明如下所示。

- ☑ sUriMatcher.addURI(Diary.AUTHORITY, "diaries", sUriMatcher.match(uri))：如果 Uri 是 content://com.shiyongcontentprovider.diarycontentprovider/diaries/，那么 sUriMatcher.match(uri) 的返回值就是 DIARIES，如上述代码中的 sUriMatcher.addURI(Diary.AUTHORITY, "diaries",DIARIES)。
- ☑ sUriMatcher.addURI(Diary.AUTHORITY, "diaries/#", DIARY_ID)：如果 Uri 是 content://com.shiyongcontentprovider.diarycontentprovider/diaries/id（其中后边的 id 是一个数字），那么 sUriMatcher.match(uri)的返回值就是 DIARY_ID。

通过 UriMatcher 类可以很方便地判断一个 Uri 的类型，特别是判断这个 Uri 是对单个数据的请求，还是对全部数据的请求。

（6）继续编写文件 DiaryContentProvider.java，开始编写抽象方法，具体实现流程如下所示。

① 编写插入方法 insert()，具体代码如下所示。

```
@Override
 public Uri insert(Uri uri, ContentValues initialValues) {
  if (sUriMatcher.match(uri) != DIARIES) {
  throw new IllegalArgumentException("Unknown URI " + uri);
  }
   ContentValues values;
  if (initialValues != null) {
  values = new ContentValues(initialValues);
  } else {
  values = new ContentValues();
  }
   if (values.containsKey(Diary.DiaryColumns.CREATED) == false) {
  values.put(Diary.DiaryColumns.CREATED, getFormateCreatedDate());
  }
   if (values.containsKey(Diary.DiaryColumns.TITLE) == false) {
  Resources r = Resources.getSystem();
  values.put(Diary.DiaryColumns.TITLE, r
  .getString(Android.R.string.untitled));
  }
   if (values.containsKey(Diary.DiaryColumns.BODY) == false) {
  values.put(Diary.DiaryColumns.BODY, "");
  }
   SQLiteDatabase db = mOpenHelper.getWritableDatabase();
  long rowId = db.insert(DIARY_TABLE_NAME, DiaryColumns.BODY, values);
  if (rowId > 0) {
  Uri diaryUri= ContentUris.withAppendedId(   Diary.DiaryColumns.CONTENT_URI, rowId);
  return diaryUri;
  }
   throw new SQLException("Failed to insert row into " + uri);
}
```

☑ sUriMatcher.match(uri) != DIARIES：判断传进来的 Uri，如果此 Uri 不是 DIARIES 类型，那么就是一个非法的 Uri。

☑ SQLiteDatabase db = mOpenHelper.getWritableDatabase()：得到一个 SQLiteDatabase 的实例。

☑ db.insert(DIARY_TABLE_NAME, DiaryColumns.BODY, values)：插入一条记录到数据库中。

方法 insert()返回的是一个 Uri，而不是一个记录的 id，所以，需要把记录的 id 构造成一个 Uri:Uri diaryUri=ContentUris.withAppendedId(Diary.DiaryColumns.CONTENT_URI,rowId).withAppendedId()方法，下面详细介绍。

ContentUris 是 content Uri 的一个辅助类，其中有如下两个常用的方法。

➢ public static Uri withAppendedId(Uri contentUri, long id)：负责把 id 和 contentUri 连接成一个新的 Uri。本实例中用法如下。

```
ContentUris.withAppendedId(Diary. DiaryColumns.CONTENT_URI, rowed)
```

如果 rowId 为 100，则现在的 Uri 内容如下。

```
content://com.example124.diarycontentprovider/diaries/100
```

➢ public static long parseId(Uri contentUri)：负责把 content Uri 后边的 id 解析出来，例如此处的 content Uri 是 content://com.shiyongcontentprovider.diarycontentprovider/diaries/100，那么这个函数的返回值就是 100。

② 编写删除方法 delete()，具体实现代码如下所示。

```
public int delete(Uri uri, String where, String[ ] whereArgs) {
  SQLiteDatabase db = mOpenHelper.getWritableDatabase();
  String rowId = uri.getPathSegments().get(1);
  return db .delete(DIARY_TABLE_NAME, DiaryColumns._ID + "=" + rowId, null);
  }
```

☑ rowId = uri.getPathSegments().get(1)：用于得到 rowId 的值。

☑ getPathSegments()方法：得到一个 String 的 List，在本实例中 uri.getPathSegments().get(1)为 rowId，如果是 uri.getPathSegments().get(0)，则值为 diaries。

☑ db.delete(DIARY_TABLE_NAME, DiaryColumns._ID + "=" + rowId, null)：是标准的 SQLite 删除操作。第一个参数为数据表的名字，第二个参数相当于 SQL 语句中的 where 部分。

③ 编写更新数据的方法 update()，具体实现代码如下所示。

```
@Override
public int update(Uri uri, ContentValues values, String where,
          String[ ] whereArgs) {
     SQLiteDatabase db = mOpenHelper.getWritableDatabase();
     String rowId = uri.getPathSegments().get(1);
     return db.update(DIARY_TABLE_NAME, values, DiaryColumns._ID + "="
               + rowId, null);
}
```

通过上述代码，先得到 SQLiteDatabase 的实例，然后得到 rowId，最后再调用 db.update(DIARY_TABLE_NAME, values, DiaryColumns._ID + "="+ rowId, null)语句执行更新操作。

④ 编写查询一条数据的方法 query()，具体代码如下所示。

```
@Override
public Cursor query(Uri uri, String[ ] projection, String selection,
          String[ ] selectionArgs, String sortOrder) {
     SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
```

```
        switch (sUriMatcher.match(uri)) {
        case DIARIES:
            qb.setTables(DIARY_TABLE_NAME);
            break;
        case DIARY_ID:
            qb.setTables(DIARY_TABLE_NAME);
            qb.appendWhere(DiaryColumns._ID + "="
                    + uri.getPathSegments().get(1));
            break;
        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
        }
        String orderBy;
        if (TextUtils.isEmpty(sortOrder)) {
            orderBy = Diary.DiaryColumns.DEFAULT_SORT_ORDER;
        } else {
            orderBy = sortOrder;
        }
        SQLiteDatabase db = mOpenHelper.getReadableDatabase();
        Cursor c = qb.query(db, projection, selection, selectionArgs, null,
                null, orderBy);
        return c;
    }
```

关于对上述代码的具体说明如下。

☑ SQLiteQueryBuilder：是一个构造 SQL 查询语句的辅助类。

☑ sUriMatcher.match(uri)：根据返回值可以判断这次查询请求时，是请求全部数据还是某个 id 的数据。如果返回值是 DIARIES，那么只需要执行 qb.setTables(DIARY_TABLE_NAME)语句即可；如果返回值是 DIARY_ID，那么还需要将 where 部分的参数设置进去，代码为 qb.appendWhere(DiaryColumns._ID + "="+ uri.getPathSegments().get(1))。

☑ SQLiteDatabase db = mOpenHelper.getReadableDatabase()：得到一个可读的 SQLiteDatabase 实例。

☑ Cursor c = qb.query(db, projection, selection, selectionArgs, null,null, orderBy)：类似于一个标准的 SQL 查询，但是此查询是 SQLiteQueryBuilder 发起的，而不是 SQLiteDatabase 直接发起的，所以在参数方面略有不同，此函数的定义格式如下所示。

```
Query(SQLiteDatabase db, String[ ] projectionIn, String selection, String[ ] selectionArgs, String groupBy, String
having, String sortOrder, String limit)
```

各个参数的具体说明如下所示。

- 第 1 个参数为要查询的数据库实例。
- 第 2 个参数是一个字符串数组，每一项代表了需要返回的列名。
- 第 3 个参数相当于 SQL 语句中的 where 部分。
- 第 4 个参数是一个字符串数组，每一项依次替代在第 3 个参数中出现的问号（?）。
- 第 5 个参数相当于 SQL 语句中的 groupby 部分。
- 第 6 个参数相当于 SQL 语句中的 having 部分。
- 第 7 个参数描述如何进行排序。
- 第 8 个参数相当于 SQL 当中的 limit 部分，控制返回的数据的个数。

在 DiaryContentProvider 类中还有两个重要的方法，分别是 getType()和 getFormateCreatedDate()。

后者的功能是根据时间得到一个特定格式的字符串。前者是一个必须要重写的方法，重写 getType()方法的主要代码如下所示。

```
public String getType(Uri uri) {
  switch (sUriMatcher.match(uri)) {
  case DIARIES:
  return DiaryColumns.CONTENT_TYPE;
   case DIARY_ID:
  return DiaryColumns.CONTENT_ITEM_TYPE;
   default:
  throw new IllegalArgumentException("Unknown URI " + uri);
  }
  }
```

此方法用于返回一个所给 Uri 的指定数据的 MIME 类型。其返回值如果以 vnd.Android. cursor.item 开头，那么就代表这个 Uri 指定的是单条数据。如果是以 vnd.Android.cursor.dir 开头，则说明这个 Uri 指定的是全部数据。

（7）单击 Menu 按钮后会执行如下代码。

```
@Override public boolean onCreateOptionsMenu(Menu menu) {
  super.onCreateOptionsMenu(menu);
  menu.add(0, MENU_ITEM_INSERT, 0, R.string.menu_insert);
  return true;
  }
```

执行后单击 Menu 按钮会弹出“添加日记”选项，如图 6-16 所示。单击“添加日记”选项后进入如图 6-17 所示界面。

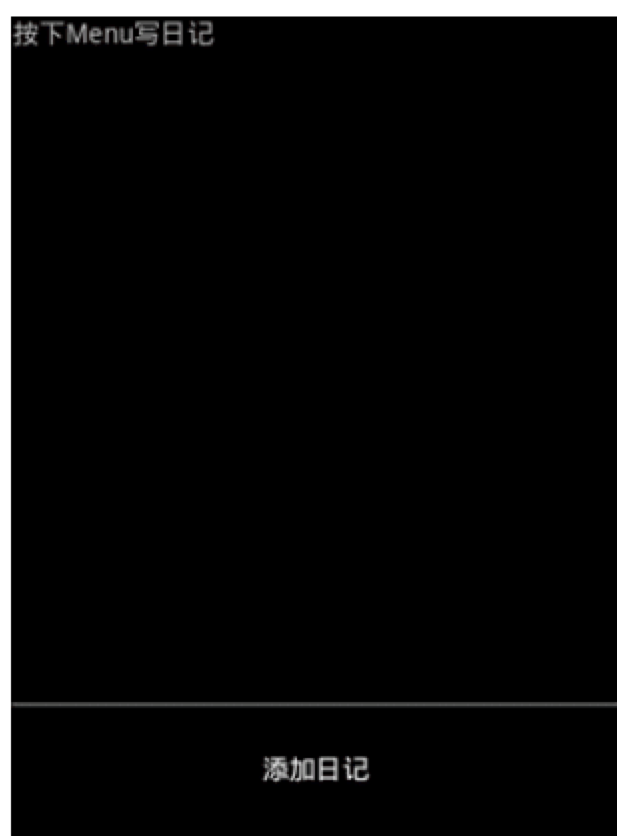


图 6-16　单击 Menu 按钮后的运行界面

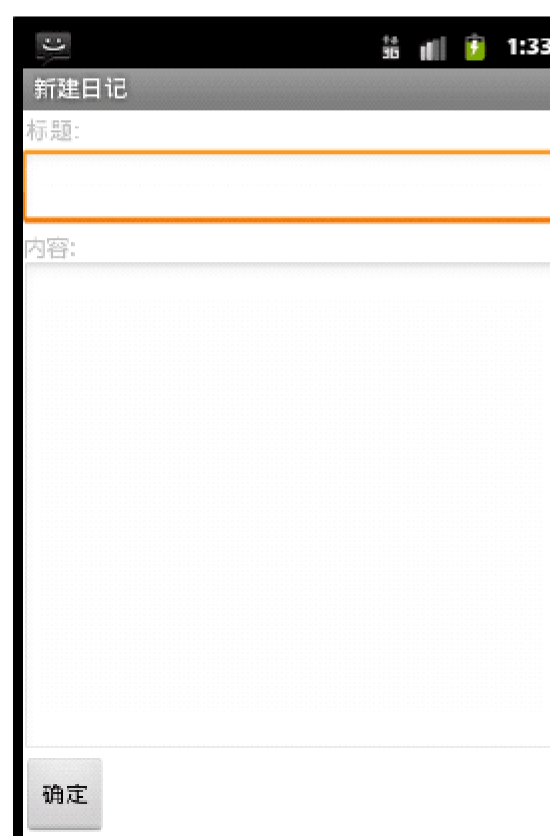


图 6-17　添加日记界面

（8）在文件 ActivityDiaryEditor.java 中定义了 ActivityDiaryEditor 类，其中有两种方法可进入日记本程序界面。一种是通过新建日记进入此界面，另一种是经过编辑日记进入此日记运行界面。下面是在文件 ActivityDiaryEditor 中使用的方法 onCreate()，其代码如下所示。

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setTheme(android.R.style.Theme_Black);
         final Intent intent = getIntent();
```

```
        final String action = intent.getAction();
        setContentView(R.layout.diary_edit);

        mTitleText = (EditText) findViewById(R.id.title);
        mBodyText = (EditText) findViewById(R.id.body);
        confirmButton = (Button) findViewById(R.id.confirm);

        if (EDIT_DIARY_ACTION.equals(action)) {                //编辑日记
            mState = STATE_EDIT;
            mUri = intent.getData();
            mCursor = managedQuery(mUri, PROJECTION, null, null, null);
            mCursor.moveToFirst();
            String title = mCursor.getString(1);
            mTitleText.setTextKeepState(title);
            String body = mCursor.getString(2);
            mBodyText.setTextKeepState(body);
            setResult(RESULT_OK, (new Intent()).setAction(mUri.toString()));
            setTitle("编辑日记");
        } else if (INSERT_DIARY_ACTION.equals(action)) {       //新建日记
            mState = STATE_INSERT;
            setTitle("新建日记");
        } else {
            Log.e(TAG, "no such action error");
            finish();
            return;
        }
        confirmButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                if (mState == STATE_INSERT) {
                    insertDiary();
                } else {
                    updateDiary();
                }
                Intent mIntent = new Intent();
                setResult(RESULT_OK, mIntent);
                finish();
            }
        });
    }
```

在上述代码中，通过 getIntent()得到启动当前 Activity 的 Intent。

通过 intent.getAction()得到该 Intent 的动作（Action）。在此操作中，当前的 Activity 是通过单击之前的“新建日记”按钮进来的，所以很有必要了解文件 example124.java 中方法 onOptionsItemSelected()的动作（Action）是如何设置的，其代码如下所示。

```
Intent intent0 = new Intent(this, ActivityDiaryEditor.class);
intent0.setAction(ActivityDiaryEditor.INSERT_DIARY_ACTION);
intent0.setData(getIntent().getData());
startActivity(intent0);
```

从上述代码可以看到，通过语句 intent0.setAction(ActivityDiaryEditor.INSERT_DIARY_ACTION)设置的 action 是 ActivityDiaryEditor.INSERT_DIARY_ACTION 实现的。

在 ActivityDiaryEditor 的方法 onCreate()中，当动作（Action）为 INSERT_DIARY_ACTION 时执行 mState = STATE_INSERT，也就是标记当前的状态为插入状态；如果当前的动作（Action）是 EDIT_DIARY_ACTION，那么当前就是编辑状态，即 mState = STATE_EDIT。

当运行程序填充数据后单击“确定”按钮，执行单击监听方法 onClick()，此方法可以根据不同的状态执行插入或者更新数据的操作。

写完日记并单击“确定”按钮后，程序将执行插入操作，具体代码如下所示。

```
private void insertDiary() {
        String title = mTitleText.getText().toString();
        String body = mBodyText.getText().toString();
        ContentValues values = new ContentValues();
        values.put(Diary.DiaryColumns.CREATED, DiaryContentProvider
                .getFormateCreatedDate());
        values.put(Diary.DiaryColumns.TITLE, title);
        values.put(Diary.DiaryColumns.BODY, body);
        getContentResolver().insert(Diary.DiaryColumns.CONTENT_URI, values);
}
```

在上述代码中，首先通过方法 getText()读取编辑框中的数据，然后将要插入的数据都放到一个 ContentValues 实例中。调用 getContentResolver()得到当前应用的一个 ContentResolver 的实例。

（9）当单击“确定”按钮后，出现如图 6-18 所示的界面。按向下方向键将焦点移动到这条数据上，然后单击 Menu 按钮会出现如图 6-19 所示界面。



图 6-18　已经插入的数据



图 6-19　单击 Menu 按钮界面

图 6-19 所示的界面和刚才介绍的单击 Menu 按钮出现的界面一样，此界面的生成代码如下所示。

```
/*在每一次 menu 生成前都会调用该方法，在此方法中可以动态修改生成的 menu*/
public boolean onPrepareOptionsMenu(Menu menu) {
        super.onPrepareOptionsMenu(menu);
        final boolean haveItems = getListAdapter().getCount() > 0;
        if (haveItems) {
                /*如果选中一个 Item*/
                if (getListView().getSelectedItemId() > 0) {
                        menu.removeGroup(1);
                        Uri uri = ContentUris.withAppendedId(getIntent().getData(),
                                        getSelectedItemId());
                        Intent intent = new Intent(null, uri);
```

```
                menu.add(1, MENU_ITEM_EDIT, 1, "编辑内容").setIntent(intent);
                menu.add(1, MENU_ITEM_DELETE, 1, "删除当前日记");
            }
        }else{
            menu.removeGroup(1);
        }
        return true;
    }
```

在上述代码中，和 ListView 相关联的 ListAdapter 中元素的个数大于 0 时 haveItems 为真，否则为假。具体说明如下。

☑ menu.removeGroup(1)：如果 Menu 中有分组为一组的子项，则先全部删除。

☑ getIntent().getData()：得到的 Uri 是 DiaryColumns.CONTENT_URI。

☑ ContentUris.withAppendedId(getIntent().getData(),getSelectedItemId())：生成一个和 ListView 中获得焦点这一项的数据的 Uri。

☑ menu.add(1, MENU_ITEM_EDIT, 1, "编辑内容").setIntent(intent)：在 menu 当中增加了一项“编辑”，并且这一项和一个 intent 关联，这个 intent 主要用于携带数据，此处携带的就是一个 Uri 的数据，在下面的 onOptionsItemSelected 函数中会用到。

☑ menu.add(1, MENU_ITEM_DELETE, 1, "删除当前日记")：用于增加另外一项。

如果 haveItems 为假，也就是当前和这个 ListView 相关联的 ListAdapter 中元素的个数为 0，即数据显示在 ListView 上时，单击 Menu 按钮，如果 menu 中还有第一分组的子项，则将其删除，实现语句为 menu.removeGroup(1)。

当单击 Menu 按钮时，在 Activity 中回调的函数可能有如下两个。

☑ onOptionsItemSelected()：此函数只在第一次在当前应用当中单击 Menu 按钮时回调，以后再不回调。

☑ onPrepareOptionsMenu()：此函数在每次单击 Menu 按钮后显示 menu 时被系统回调，每次 menu 显示前都会回调此函数。一般根据条件改变 menu 显示的逻辑都放在该函数中。

当单击 Menu 按钮时弹出 3 个按钮，单击其中的某一个按钮会触发 Android 系统回调 onOptionsItemSelected 函数，此函数实现代码如下所示。

```
@Override public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {                                    //插入一条数据
  case MENU_ITEM_INSERT:
  Intent intent0 = new Intent(this, ActivityDiaryEditor.class);
  intent0.setAction(ActivityDiaryEditor.INSERT_DIARY_ACTION);
  intent0.setData(getIntent().getData());
  startActivity(intent0);
  return true;                                                   //编辑当前数据内容
  case MENU_ITEM_EDIT:
  Intent intent = new Intent(this, ActivityDiaryEditor.class);
  intent.setData(item.getIntent().getData());   intent.setAction(ActivityDiaryEditor.EDIT_DIARY_ACTION);
  startActivity(intent);   return true;               //删除当前数据
  case MENU_ITEM_DELETE:
  Uri uri = ContentUris.withAppendedId(getIntent().getData(),
  getListView().getSelectedItemId());
  getContentResolver().delete(uri, null, null);
  renderListView();
```

```
    }
  return super.onOptionsItemSelected(item);
  }
```

当用户单击“添加日记”按钮后，程序新建一个跳转 Activity 的 intent0，并且设置了 Action 和 data，这两部分在程序跳转到 ActivityDiaryEditor 后会用到。

当用户单击“编辑内容”按钮后，程序也新建了一个跳转 Activity 的 intent，并且设置了 Action 和 data。同样，这两部分在程序跳转到 ActivityDiaryEditor 后会用到。需要注意的是，通过 item.getIntent().getData()得到了所需要的 Uri。

当用户单击“删除当前日记”按钮后，程序通过 ContentUris.withAppendedId(getIntent().getData(), getListView().getSelectedItemId())先得到需要删除数据的 Uri，然后得到当前的 ContentResolvor，再调用其 delete()方法进行删除。当删除数据后，调用 renderListView 函数对 ListView 进行及时刷新。

6.6 存储当前用户的信息

实例 088	保存用户信息
源码路径	光盘:\daima\088
视频路径	光盘:\视频\088
实例必备	088.XML 文件的形式保存数据.pdf

6.6.1 实例说明

在编程时经常需要保存一些用户设置的信息，例如，是否记住密码、显示的字体大小等。在 Android 应用中，可以使用 SharedPreferences 实现此功能，SharedPreferences 是一个轻量级的存储类，特别适合用于保存软件配置参数。SharedPreferences 是用 XML 文件存放数据，文件存放在\data\data\package name\shared_prefs 目录下。

6.6.2 具体实现

（1）编写文件 SharedPreferencesUtil.java，在此定义了一个工具类，在执行保存操作时要使用 commit()。另外，因为 SharedPreferences 本身就是以 XML 保存文件的，所以命名时无须再添加.xml 后缀。文件 SharedPreferencesUtil.java 的主要代码如下所示。

```
public class SharedPreferencesUtil {
	private SharedPreferences sp;
	private Editor editor;
	private final static String SP_NAME = "mydata";
	private final static int MODE = Context.MODE_WORLD_READABLE
			+ Context.MODE_WORLD_WRITEABLE;
	public SharedPreferencesUtil(Context context) {
		sp = context.getSharedPreferences(SP_NAME, MODE);
		editor = sp.edit();
	}
```

```
    public boolean save(String key, String value) {
        editor.putString(key, value);
        //注意添加 commit
        return editor.commit();
    }
    public String read(String key) {
        String str = null;
        str = sp.getString(key, null);
        return str;
    }
}
```

（2）编写主程序文件，主要代码如下所示。

```
    /**
     * 给控件初始化
     */
    public void init() {
        util = new SharedPreferencesUtil(this);
        saveBtn = (Button) findViewById(R.id.save_btn);
        readBtn = (Button) findViewById(R.id.read_btn);
        inputEv = (EditText) findViewById(R.id.input_et);
        showEv = (EditText) findViewById(R.id.showinfo_et);
        //设置监听
        setListener();
    }

    /**
     * 给 Button 加监听事件
     */
    public void setListener() {
        saveBtn.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                boolean b = util.save("mykey", inputEv.getText().toString());
                String msg;
                if (b) {
                    msg = "保存成功";
                } else {
                    msg = "保存失败";
                }
                Toast.makeText(example125.this, msg,
                        Toast.LENGTH_SHORT).show();
            }
        });

        readBtn.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                System.out.println("read...");
                String value = util.read("mykey");
                showEv.setText(value);
            }
        });
    }
```

```
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                init();
        }
}
```

执行后的效果如图 6-20 所示，在文本框中输入文本，先单击“保存”按钮，然后单击“读取”按钮后会在下方文本框中显示输入的数据，如图 6-21 所示，单击“保存”按钮。



图 6-20 执行效果



图 6-21 在下方文本框显示读取的数据

6.7 使用文件保存数据

实例 089	使用文件保存数据
源码路径	光盘:\daima\089
视频路径	光盘:\视频\089
实例必备	089.方法 openFileOutput().pdf

6.7.1 实例说明

在 Android 手机系统中，可以使用文件来保存数据。与在 Java 中实现 I/O 的程序类似，在 Android 中提供了 openFileInput()和 openFileOuput()方法读取设备上的文件，请读者看下面的代码。

```
String FILE_NAME = "tempfile.tmp";                          //确定要操作文件的文件名
  FileOutputStream fos = openFileOutput(FILE_NAME, Context.MODE_PRIVATE); //初始化
  FileInputStream fis = openFileInput(FILE_NAME);           //创建写入流代码解释
```

在上述代码中的两个方法只支持读取该应用目录下的文件，读取非其自身目录下的文件将会抛出异常。需要注意的是，如果调用 FileOutputStream()时指定的文件不存在，Android 会自动创建。另外，在默认情况下，写入时会覆盖原文件内容，如果想把新写入的内容附加到原文件内容后，则可以指定其模式为 Context.MODE_APPEND。

另外，在 CSDN 论坛中经常看到关于存储方式持久性的问题，询问关机后能够继续存在的存储方式。其实关机后依然存在的存储方式是 File 文件存储、SharedPreference 存储、SQLite 存储和 ContentProvider 存储。这 4 种存储方式的主要特点如下所示。

☑ File 文件存储：主要存储大型文件，但需要 sdcard 中有相应空间，例如，存储一个二进制文

件。操作方式与普通 Java 相似，即打开一个 FileInputStream/FileOutPutStream，转成 InputStream/outPutStream，然后读/写字节。

☑ SharedPrefreence 存储：主要用来存储简单数据类型，不能存文件。例如，第一次登录 QQ 后可以保存账号和密码（用户选择记住密码），下次用户再登录时直接进入，不需要用户再输入。

☑ SQLite 存储：是小型数据库，主要用来存记录表格。例如，存多个玩家的积分排行榜，需要有 id、score、level 等字段组成的 N 行表格。

☑ ContentProvider 存储：又称内容提供器，提供一种实现两个不相关的应用程序之间进行通信的方式，例如，程序 A 在指定的 ContentProvider 中存储下一个数据，程序 B 可以读取。

6.7.2　具体实现

（1）编写文件 MainActivity.java，定义保存文件并读取文件内容的方法，主要代码如下所示。

```
public class MainActivity extends Activity {
	private EditText writeET;
	private Button writeBtn;
	private TextView contentView;
	public static final String FILENAME = "setting.set";

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		writeET = (EditText) findViewById(R.id.write_et);
		writeBtn = (Button) findViewById(R.id.write_btn);
		contentView = (TextView) findViewById(R.id.contentview);
		writeBtn.setOnClickListener(new OperateOnClickListener());
	}

	class OperateOnClickListener implements OnClickListener {
		@Override
		public void onClick(View v) {
			writeFiles(writeET.getText().toString());
			contentView.setText(readFiles());
			System.out.println(getFilesDir());
		}
	}

	//保存文件内容
	private void writeFiles(String content) {
		try {
			//打开文件获取输出流，文件不存在则自动创建
			FileOutputStream fos = openFileOutput(FILENAME,
					Context.MODE_PRIVATE);
			fos.write(content.getBytes());
			fos.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
```

```
    //读取文件内容
    private String readFiles() {
        String content = null;
        try {
            FileInputStream fis = openFileInput(FILENAME);
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            byte[ ] buffer = new byte[1024];
            int len = 0;
            while ((len = fis.read(buffer)) != -1) {
                baos.write(buffer, 0, len);
            }
            content = baos.toString();
            fis.close();
            baos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return content;
    }
}
```

（2）编写文件 FilesUtil.java，分别实现保存文件和读取文件内容的功能，主要代码如下所示。

```
public class FilesUtil {
    /**保存文件内容，fileName 表示文件名称，content 表示内容*/
    private void writeFiles(Context c, String fileName, String content, int mode)
            throws Exception {
        //打开文件获取输出流，文件不存在则自动创建
        FileOutputStream fos = c.openFileOutput(fileName, mode);
        fos.write(content.getBytes());
        fos.close();
    }

    /*读取文件内容，return 表示返回文件内容*/
    private String readFiles(Context c, String fileName) throws Exception {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        FileInputStream fis = c.openFileInput(fileName);
        byte[ ] buffer = new byte[1024];
        int len = 0;
        while ((len = fis.read(buffer)) != -1) {
            baos.write(buffer, 0, len);
        }
        String content = baos.toString();
        fis.close();
        baos.close();
        return content;
    }
}
```

执行后的效果如图 6-22 所示，在文本框中输入信息并单击 Write 按钮后将信息写入并保存到文件，并在按钮下方显示输入的信息，如图 6-23 所示。



图 6-22　执行效果

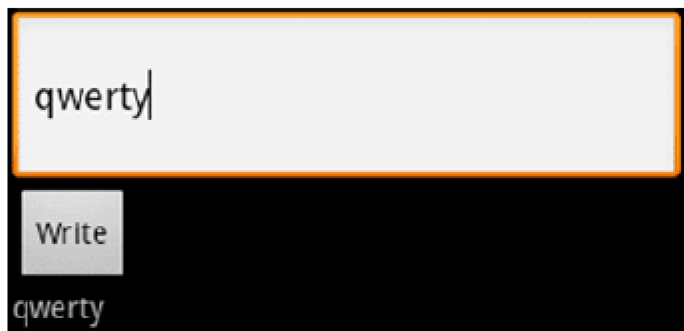


图 6-23　保存输入的信息

6.8　使用 SD 卡保存图片

实例 090	将网上图片保存在 SD 卡中并显示出来
源码路径	光盘:\daima\090
视频路径	光盘:\视频\090
实例必备	090.总结数据存储方式.pdf

6.8.1　实例说明

在现实开发应用中，经常需要使用网络中的资源。在本实例中，首先下载获取远程网络中的图片，然后将下载的图片保存在 SD 卡中，最后将 SD 卡中的图片从屏幕中显示出来。

6.8.2　具体实现

（1）编写文件 Activity01.java，监听单击“下载”按钮事件，单击后下载指定地址的图片，并且定义方法 GetNetBitmap()来获取网络中的图片，主要代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

mEditText=(EditText)findViewById(R.id.edit);
btn1=(Button)findViewById(R.id.download_Btn);                    //“下载”按钮
btn2=(Button)findViewById(R.id.show_pic);                        //显示图按钮

//为显示按钮设置监听，保存 URL 的数据
btn2.setOnClickListener(new Button.OnClickListener() {

            public void onClick(View v) {
                Intent intent=new Intent();
                    intent.setClass(Activity01.this, A1.class);
                    startActivity(intent);
                    Activity01.this.finish();                     //结束
            }
        });
//下载图片
btn1.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
```

```
    //判断是否可以对 SD 卡进行操作
    if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
    String url=mEditText.getText().toString().trim();
    String fileName=url.substring(url.lastIndexOf('/')+1,url.length());          //提取下载图片的文件名
    Bitmap bitmap=GetNetBitmap(url);                                               //得到 bitmap
    String sdCardDir = Environment.getExternalStorageDirectory()+"/hekui/";  //获取 SD 卡目录
                    File file = new File(sdCardDir,fileName);                      //在 SD 卡的目录下创建图片文件
        try //将网络上读取的图片保存到 SD 卡中
        {
            FileOutputStream out=new FileOutputStream(file);                       //为图片文件实例化输出流
            if(bitmap.compress(Bitmap.CompressFormat.PNG, 100, out))   //保存图片
            {
            out.flush();
            Log.v(TAG,"Success");
            out.close();
             info=(TextView)findViewById(R.id.textView1);
            info.setText(fileName+"  下载成功！！ ");

            }
        }
        catch (FileNotFoundException e)
        {
            Log.v(TAG,"文件没发现！！ ");
            e.printStackTrace();
            } catch (IOException e)
            {
            e.printStackTrace();
            Log.v(TAG,"数据流错误！！ ");
            }
      }
    }
});
}

//获取网络上的图片
public Bitmap GetNetBitmap(String url)
{
        URL imageUrl = null;
        Bitmap bitmap=null;
        try
        {
            imageUrl = new URL(url);
        }
        catch (MalformedURLException e)
        {
            Log.v(TAG, e.getMessage());
        }

        if (imageUrl != null) {
            try {
                HttpURLConnection conn = (HttpURLConnection)imageUrl.openConnection();
```

```
                conn.setDoInput(true);                      //设置请求的方式
                conn.connect();

                InputStream is = conn.getInputStream();       //将得到的数据转化为 inputStream
                bitmap = BitmapFactory.decodeStream(is);    //将 inputstream 转化为 Bitmap
                is.close();// 关闭数据

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        else
        {
            Log.v(TAG,"Url is Null!!");
        }
        return bitmap;
}
}
```

（2）编写文件 sdk_fileclass.java，获取 Android“下载/缓存”内容目录，主要代码如下所示。

```
public class sdk_fileclass {
    /*.getDownloadCacheDirectory()获取 Android“下载/缓存”内容目录*/
    private String dirName = Environment
        .getExternalStorageDirectory().toString()+"/hekui/";
    public String[ ] filenames = new File(dirName).list();
    public int getCount(){
        if(filenames == null)
            return 0;
        return filenames.length;
    }
    public Bitmap getImageAt(String[ ] filenames, int i){
        String path = dirName;
        if(i>=filenames.length)
            return null;
        path+=filenames[i];
        Bitmap b = BitmapFactory.decodeFile(path);
        return b;
    }
}
```

执行后的效果如图 6-24 所示，输入图片地址并单击“下载”按钮后会下载此图片，单击“显示”按钮会显示下载的图片。



图 6-24　执行效果

第 7 章　电话和短信实战

虽然当前 Android 智能手机的功能越来越强大，但是消费者购买手机的最主要目的还是接打电话和收发短信。在本章的内容中，将通过具体实例的实现流程详细讲解在 Android 系统中实现拨打电话、接听电话、发送短信和接收短信的基本用法。

7.1　实现简单的拨打电话功能

实例 091	实现一个简单的拨号程序
源码路径	光盘:\daima\091
视频路径	光盘:\视频\091
实例必备	091.Service 详解.pdf ① Service 基础 ② Service 的生命周期 ③ Service 的策略 ④ 创建 Service

7.1.1　实例说明

拨号功能离不开 IntentFilter，在本实例中，使用一个 Intent 打开电话拨号程序，Intent 的行为是 ACTION_DIAL，同时在 Intent 中传递被呼叫人的电话号码。整个程序的实现分为 3 个阶段：第 1 阶段完成向固定电话拨号的工作，用户不能自由输入希望通话的电话号码；第 2 阶段负责进一步完善用户界面，让用户可以自由输入电话号码后再拨号；第 3 阶段加入 IntentFilter，使用户可以通过硬键盘拨号键启动程序。

7.1.2　具体实现

（1）编写文件 main.xml，在界面中加入一个 Button 控件和一个<TextView>标签，加入新的<Button>标签。把 Button 的 id 设置为 button_id，同时将 Button 显示在界面上的文字设置为 res\string.xml\下的 Button，打开 res\string.xml，把 Button 的内容设置为“拨号”。

（2）编写文件 Dialer.java 代码，在其中定位“拨号”按钮。要加入对“拨号”按钮的响应，首先通过 findViewById()方法获得该按钮对象的引用，然后加入对“拨号”按钮按键动作的响应，为“拨号”按钮对象调用 setOnClickListener()方法，设置单击事件监听器。文件 Dialer.java 的主要实现代码如下所示。

```
public class Dialer extends Activity {
/** Called when the Activity is first created.*/
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new Button.OnClickListener() {
@Override
public void onClick(View b) {
Intent i = new Intent(Intent.ACTION_DIAL, Uri.parse("tel://13800138000"));
startActivity(i);
}
});
}
}
```

（3）进一步完善第 1 阶段中的成果，使得用户能输入电话号码。由于用户输入的可能不是一个有效的电话号码，所以程序还需要对用户输入的字符串进行判断，呼叫有效号码，如果是无效号码，则提示用户重新输入。

① 修改用户界面，加入获取用户输入的 EditText 控件。在 Button 控件前加入一个 EditText 控件用于获取用户输入的电话号码，设置其 id 引用名为 phonenumber_id。

```
<EditText android:id = "@+id/phonenumber_id"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 />
 <Button android:id = "@+id/button_id"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content" android:text="@string/button"
 />
```

② 获得 EditText 对象的引用，代码如下所示。

```
final EditText phoneNumber = (EditText)findViewById(R.id.phonenumber_id);
```

③ 在回调方法 onClick()中加入对电话号码有效性的判断和处理，具体代码如下所示。

```
@Override   public void onClick(View b) {
  String callee = phoneNumber.getText().toString();
  if (PhoneNumberUtils.isGlobalPhoneNumber(callee)){
  Intent i = new Intent(Intent.ACTION_DIAL,   Uri.parse(tel"://"+ callee));   startActivity(i);
  } else {
  Toast.makeText(TinyDialer.this, R.string.notify_incorrect_phonenumber,
  Toast.LENGTH_LONG).show();
  }
  }
```

在上述代码中，使用 android.telephony.PhoneNumberUtils 包中的方法 isGlobalPhoneNumber()来判断电话号码的有效性，在 Android 中准备了很多类似的方法可以简化程序员的工作量，用好这些方法能够帮助程序员轻松快速地完成工作。另外，应使用 Toast 类输出无效电话号码提示。

至此，整个实例编写完毕。文件 Dialer.java 的完整代码如下所示。

```
public class Dialer extends Activity {
    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);

            final EditText phoneNumber = (EditText) findViewById(R.id.phonenumber_id);
            final Button button = (Button) findViewById(R.id.button_id);
            button.setOnClickListener(new Button.OnClickListener()
                {
                        @Override
                        public void onClick(View b) {
                            String callee = phoneNumber.getText().toString();
                            if (PhoneNumberUtils.isGlobalPhoneNumber(callee)){
                                //Intent i = new Intent(Intent.ACTION_DIAL, Uri.parse("tel://" + callee));
                                Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel://" + callee));
                                startActivity(i);
                            } else {
                                Toast.makeText(Dialer.this, R.string.notify_incorrect_phonenumber,
                                    Toast.LENGTH_LONG).show();
                            }
                        }
                }
            );
        }
}
```

执行后的效果如图 7-1 所示。在文本框中输入电话号码并单击“拨打号码”按钮后将自动切换到 Android 自带的拨号程序，同时所拨打的电话号码将会显示在屏幕上，如图 7-2 所示。

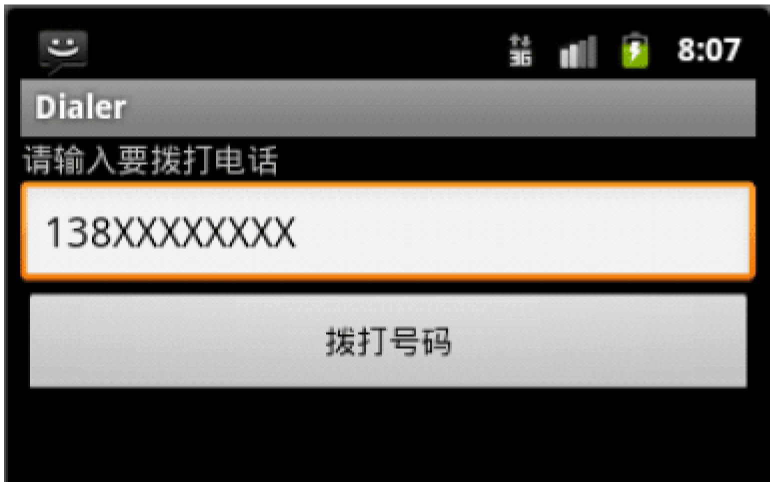


图 7-1 执行效果

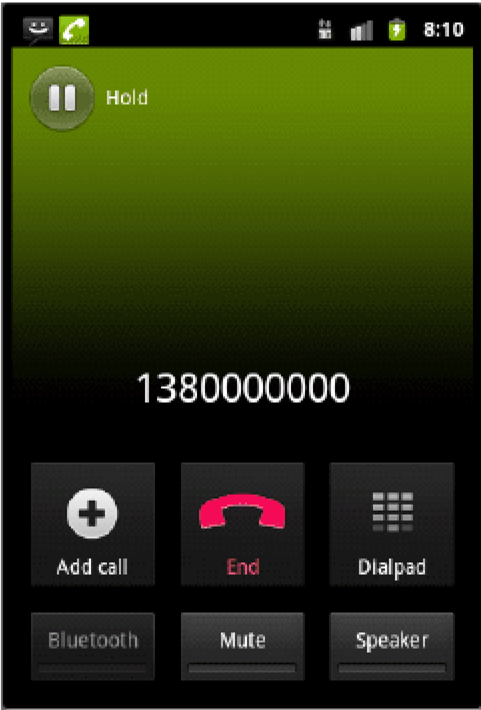


图 7-2 Android 自带拨号程序界面

7.2 发送一则短信息

实例 092	实现发送短信功能
源码路径	光盘:\daima\092
视频路径	光盘:\视频\092
实例必备	092.使用 Service.pdf

7.2.1　实例说明

和电话拨号程序一样，短信功能也是任何一款手机不可或缺的基本应用，是使用频率最高的程序之一。在本实例中，不是简单地使用 Intent 激活 Android 自带的短信程序，而是使用 SmsManager 类完成发送短信的功能。笔者希望借此实例抛砖引玉，帮助读者进一步理解 Android 中常见类的用法。

7.2.2　具体实现

（1）编写布局文件 main.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="输入短信号码"
        />
    <EditText
        android:id="@+id/txtPhoneNo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="短信"
        />
    <EditText
        android:id="@+id/txtMessage"
        android:layout_width="fill_parent"
        android:layout_height="150px"
        android:gravity="top"
        />
    <Button
        android:id="@+id/btnSendSMS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="发送短信"
        />
</LinearLayout>
```

设计后的界面如图 7-3 所示。

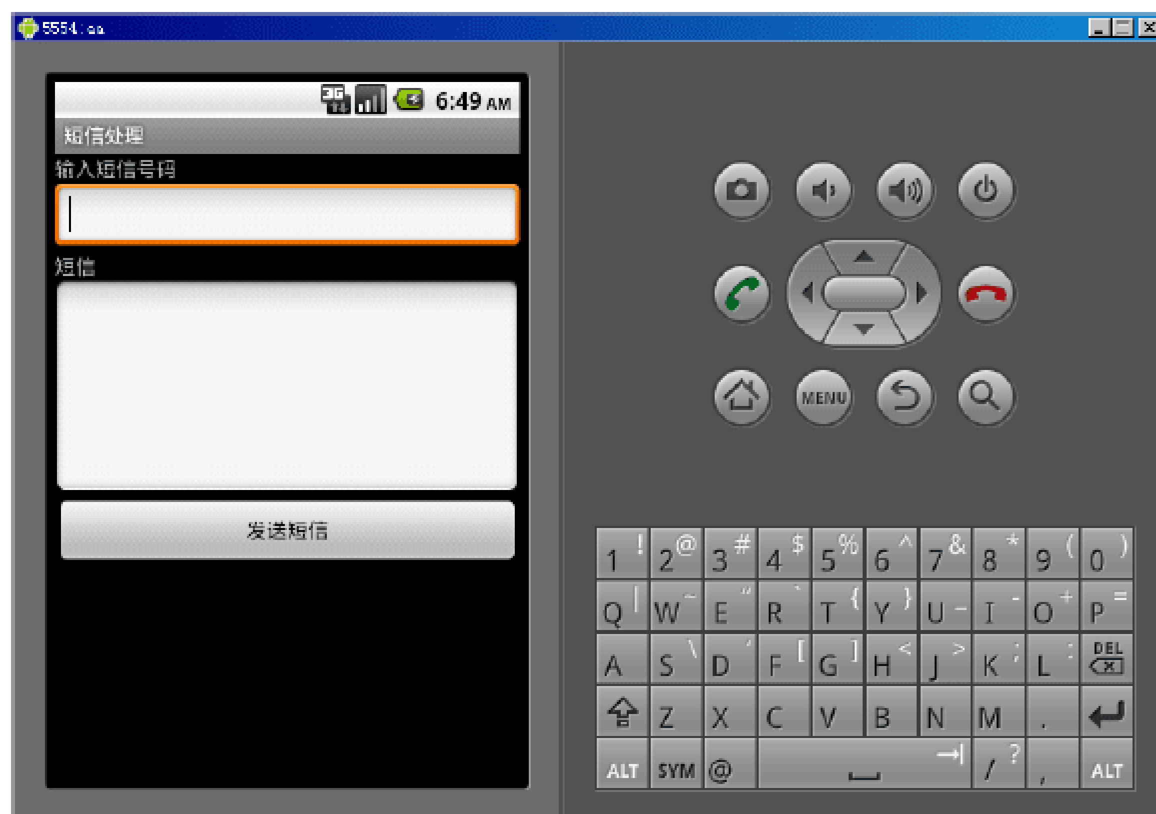


图 7-3　程序界面

（2）设置权限，因为项目程序需要使用发送短信功能，根据对 AndroidManifest.xml 的描述，在此需要在该文件中声明程序的权限。因此这里需要加入 TinySMS 发送短信的权限的声明，主要代码如下所示。

```
//发送短信权限
<uses-permission android:name="android.permission.SEND_SMS" />
```

（3）编写文件 example.java 实现发送短信处理，单击“发送短信”按钮后调用回调方法 onClick()实现发送短信的功能，具体代码如下所示。

```
        btnSendSMS.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v)
            {
                String phoneNo = txtPhoneNo.getText().toString();
                String message = txtMessage.getText().toString();
                if (phoneNo.length()>0 && message.length()>0){
                    Log.v("ROGER", "will begin sendSMS");
                        sendSMS(phoneNo, message);
                }
                else
                    Toast.makeText(SMSchuli.this,
                        "请重新输入电话号码和短信内容",
                        Toast.LENGTH_LONG).show();
            }
        });
```

TinySMS 并不是使用 Intent 激活 Android 自带的短信程序，而是直接使用了一个 sendSMS()方法。该方法的实现代码如下所示。

```
    private void sendSMS(String phoneNumber, String message)   {
        PendingIntent pi = PendingIntent.getActivity(this, 0,
            new Intent(this, SMSchuli.class), 0);
        Log.v("ROGER", "will init SMS Manager");
        SmsManager sms = SmsManager.getDefault();

        Log.v("ROGER", "will send SMS");
        sms.sendTextMessage(phoneNumber, null, message, pi, null);
    }
```

SmsManager 是在 android.telephony.gsm.SmsManager 中定义的用户管理短信应用的类。其用法有些特殊，开发人员不用直接实例化 SmsManager 类，而只需要调用静态方法 getDefault()获得 SmsManger 对象，方法 sendTextMessage()用于发送短信到指定号码。在上述代码中，使用了一个 PendingIntent 的对象，该对象指向 TinySMSActivity。因此当用户单击“发送短信”按钮之后，用户界面会重新回到 TinySMS 的初始界面。

在 Android 的模拟器中对短信或电话提供了非常方便的测试功能。用户只需要在 Windows 命令行中输入 emulator 再启动一个 Android 模拟器，即可实现两个手机间的电话或者短信的测试。需要说明的是，每个模拟器左上角的数字代表了该模拟器的电话号码。

7.3　实现按钮拨号功能

实例 093	实现一个按钮拨号程序
源码路径	光盘:\daima\093
视频路径	光盘:\视频\093
实例必备	093.与远程 Service 通信.pdf

7.3.1　实例说明

本实例演示了实现手机拨打电话的过程，首先使用了一个 EditText 用于获取输入的电话号码，单击 Button 控件后执行拨打电话操作，通过自定义的 isPhoneNumberValid(String phoneNumber)验证用户输入的是否为合法的电话号码。

在拨打电话时，首先在 AndroidManifest 中声明 uses-permission 权限，然后通过自定义的 Intent 对象、ACTION_CALL 键和 Uri.parse()方法写入用户输入的电话号码，最后通过方法 startActivity()实现拨打电话的功能。

7.3.2　具体实现

编写文件 example.java，下面开始讲解其具体实现代码。

（1）引用类和对象，声明 Button 与 EditText 对象名称，主要代码如下所示。

```
/*声明 Button 与 EditText 对象名称*/
  private Button mButton1;
  private EditText mEditText1;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    /*通过 findViewById 构造器构造 EditText 与 Button 对象*/
    mEditText1 = (EditText) findViewById(R.id.myEditText1);
    mButton1 = (Button) findViewById(R.id.myButton1);
```

（2）设置 Button 按钮对象的 OnClick 单击事件，主要代码如下所示。

```
    mButton1.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
        try
        {
          /*取得 EditText 中用户输入的字符串*/
          String strInput = mEditText1.getText().toString();
          if (isPhoneNumberValid(strInput)==true)
          {
            /*建构一个新的 Intent，运行 action.CALL 的常数与通过 URI 将字符串带入*/
            Intent myIntentDial = new
            Intent
            (
              "android.intent.action.CALL",
              Uri.parse("tel:"+strInput)
            );
            /*在 startActivity()方法中带入自定义的 Intent 对象以执行拨打电话的操作*/
            startActivity(myIntentDial);
            mEditText1.setText("");
          }
          else
          {
            mEditText1.setText("");
            Toast.makeText(
            example.this, "电话格式不正确",
            Toast.LENGTH_LONG).show();
          }
        }
        catch(Exception e)
        {
          e.printStackTrace();
        }
      }
    });
  }
```

（3）定义方法 isPhoneNumberValid()来检查字符串是否为电话号码，返回 true 表示是合法的电话号码，返回 false 表示不是合法的电话号码，主要代码如下所示。

```
  public static boolean isPhoneNumberValid(String phoneNumber)
  {
    boolean isValid = false;
    /*可接受的电话格式有
     * ^\\(?: 可以使用 “(” 作为开头
     * (\\d{3}): 紧接着 3 个数字
     * \\)?: 可以使用 “)” 继续
     * [- ]?: 在上述格式后可以使用具选择性的 “-”
     * (\\d{4}): 再紧接着 4 个数字
     * [- ]?: 可以使用具选择性的 “-” 继续
     * (\\d{4})$: 以 4 个数字结束
     *  可以比较下列数字格式:
```

```
     * (123)456-78900, 123-4560-7890, 12345678900, (123)-4560-7890
    */
    String expression = "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{5})$";
    String expression2 ="^\\(?(\\d{3})\\)?[- ]?(\\d{4})[- ]?(\\d{4})$";
    CharSequence inputStr = phoneNumber;
    /*创建 Pattern*/
    Pattern pattern = Pattern.compile(expression);
    /*将 Pattern 以参数传入 Matcher 作 Regular expression*/
    Matcher matcher = pattern.matcher(inputStr);
    /*创建 Pattern2*/
    Pattern pattern2 =Pattern.compile(expression2);
    /*将 Pattern2 以参数传入 Matcher2 作 Regular expression*/
    Matcher matcher2= pattern2.matcher(inputStr);
    if(matcher.matches()||matcher2.matches())
    {
      isValid = true;
    }
    return isValid;
  }
}
```

执行后的效果如图 7-4 所示；如果输入的电话号码不规范则输出对应的提示，如图 7-5 所示；当输入规范的号码并单击“拨打电话”按钮后显示拨打界面，实现拨打电话功能，如图 7-6 所示。



图 7-4　执行效果



图 7-5　自动显示输入数据

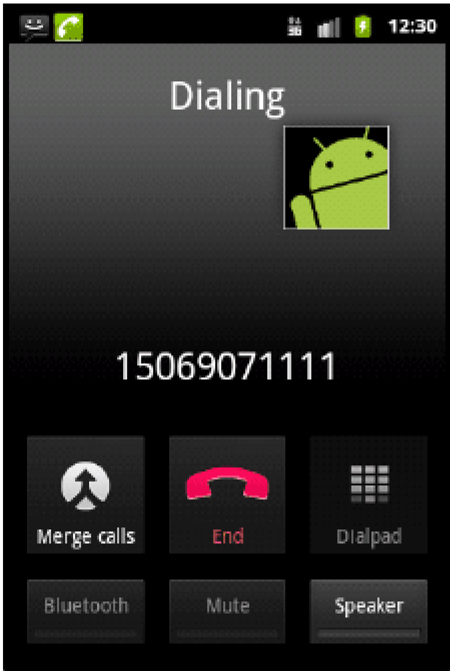


图 7-6　拨打界面

在上述代码中，使用方法 isPhoneNumberValid()检查字符串是否为电话号码，这其实是一个正则表达式功能。

7.4　实现发送短信系统

实例 094	实现一个发送短信系统
源码路径	光盘:\daima\094
视频路径	光盘:\视频\094
实例必备	094.提高 Service 优先级.pdf

7.4.1 实例说明

在本实例中，定义了两个 EditText 控件分别获取收信人电话和短信正文，并设置了判断手机号码规范化的方法，在此规定短信的字符数不能超过 70 个。在具体实现上，是通过 SmsManage 对象的方法 sendTextMessage()来完成的。在 sendTextMessage()中要传入 5 个值，分别是“收件人地址 String”“发送地址 String”“正文 String”“发送服务 PendingIntent”“送达服务 PendingIntent”。

7.4.2 具体实现

编写文件 example.java，其具体实现流程如下所示。

（1）声明一个 Button 变量用于激活发送短信处理程序，声明两个 EditText 变量用于获取输入的收信人电话号码和短信内容，主要代码如下所示。

```
/*声明一个 Button 与两个 EditText*/
private Button mButton1;
private EditText mEditText1;
private EditText mEditText2;
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  /*通过 findViewById 构造器建构 EditText1、EditText2 和 Button 对象*/
  mEditText1 = (EditText) findViewById(R.id.myEditText1);
  mEditText2 = (EditText) findViewById(R.id.myEditText2);
  mButton1 = (Button) findViewById(R.id.myButton1);
  /*将默认文字加载到 EditText 中*/
  mEditText1.setText("请输入号码");
  mEditText2.setText("请输入内容!!");
  /*设置用户单击 EditText 时做出响应*/
  mEditText1.setOnClickListener(new EditText.OnClickListener()
  {
    public void onClick(View v)
    {
      /*单击 EditText 时清空正文*/
      mEditText1.setText("");
    }
  }
  );
```

（2）定义方法 onClickListener()响应当用户单击 EditText 时的反应，主要代码如下所示。

```
  /*定义 onClickListener()让用户单击 EditText 时做出反应*/
  mEditText2.setOnClickListener(new EditText.OnClickListener()
  {
    public void onClick(View v)
    {
      /*单击 EditText 时清空正文*/
      mEditText2.setText("");
```

```
    }
  }
  );
```

（3）定义方法 onClickListener()作为用户单击 Button 时的反应，主要代码如下所示。

```
/*定义 onClickListener()让用户单击 Button 时做出反应*/
mButton1.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    /*由 EditText1 获取短信收件人电话*/
    String strDestAddress = mEditText1.getText().toString();
    /*由 EditText2 获取短信文字内容*/
    String strMessage = mEditText2.getText().toString();
    /*建构一个获取 default instance 的 SmsManager 对象*/
    SmsManager smsManager = SmsManager.getDefault();

    // TODO Auto-generated method stub
```

（4）首先检查收件人电话格式是否正确，短信字数是否超过 70 字符，然后通过 smsManager. sendTextMessage 实现发送短信处理，具体实现代码如下所示。

```
    /*检查收件人电话格式与短信字数是否超过 70 字符*/
    if(isPhoneNumberValid(strDestAddress)==true &&
       iswithin70(strMessage)==true)
    {
      try
      {
        /*
        * 两个条件都检查通过的情况下，发送短信
        * 先建构一个 PendingIntent 对象并使用 getBroadcast()广播
        * 将 PendingIntent、电话、短信文字等参数
        * 传入 sendTextMessage()方法发送短信
        */
        PendingIntent mPI = PendingIntent.getBroadcast
        (example131.this, 0, new Intent(), 0);
        smsManager.sendTextMessage
        (strDestAddress, null, strMessage, mPI, null);
      }
      catch(Exception e)
      {
        e.printStackTrace();
      }
      Toast.makeText
      (
        Example.this,"送出成功!!" ,
        Toast.LENGTH_SHORT
      ).show();
      mEditText1.setText("");
      mEditText2.setText("");
    }
    else
```

```
            {
              /*当电话格式与短信文字不符合条件时，以 Toast 提醒*/
              if (isPhoneNumberValid(strDestAddress)==false)
              { /*且字数超过 70 字符*/
                if(iswithin70(strMessage)==false)
                {
                  Toast.makeText
                  (
                    example3.this,
                    "电话号码格式错误/短信内容超过 70 字符!!!",
                    Toast.LENGTH_SHORT
                  ).show();
                }
                else
                {
                  Toast.makeText
                  (
                    Example.this,
                    "电话号码格式错误,请检查!!" ,
                    Toast.LENGTH_SHORT
                  ).show();
                }
              }
              /*字数超过 70 字符*/
              else if (iswithin70(strMessage)==false)
              {
                Toast.makeText
                (
                  Example.this,
                  "短信内容超过 70 字符，请删除部分内容!!",
                  Toast.LENGTH_SHORT
                ).show();
              }
            }
          }
        });

    }
    /*检查字符串是否为电话号码的方法，并返回 true 或 false 的判断值*/
    public static boolean isPhoneNumberValid(String phoneNumber)
    {
      boolean isValid = false;
      /*可接受的电话格式有
       * ^\\(?：可以使用“(”作为开头
       * (\\d{3})：紧接着 3 个数字
       * \\)?：可以使用“)”继续
       * [- ]?：在上述格式后可以使用具选择性的“-”
       * (\\d{3})：再紧接着 3 个数字
       * [- ]?：可以使用具选择性的“-”继续
       * (\\d{5})$：以 5 个数字结束
       *  可以比较下列数字格式：
```

```
  * (123)456-7890, 123-456-7890, 1234567890, (123)-456-7890
 */
 String expression =
 "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{5})$";

 /*可接受的电话格式有
  * ^\\(?:可以使用“(”作为开头
  * (\\d{3}):紧接着 3 个数字
  * \\)?:可以使用“)”继续
  * [- ]?:在上述格式后可以使用具选择性的“-”
  * (\\d{4}):再紧接着 4 个数字
  * [- ]?:可以使用具选择性的“-”继续
  * (\\d{4})$:以 4 个数字结束
  *  可以比较下列数字格式:
  * (02)3456-7890, 02-3456-7890, 0234567890, (02)-3456-7890
 */
 String expression2=
 "^\\(?(\\d{3})\\)?[- ]?(\\d{4})[- ]?(\\d{4})$";

 CharSequence inputStr = phoneNumber;
 /*创建 Pattern*/
 Pattern pattern = Pattern.compile(expression);
 /*将 Pattern 以参数传入 Matcher 作 Regular expression*/
 Matcher matcher = pattern.matcher(inputStr);
 /*创建 Pattern2*/
 Pattern pattern2 =Pattern.compile(expression2);
 /*将 Pattern2 以参数传入 Matcher2 作 Regular expression*/
 Matcher matcher2= pattern2.matcher(inputStr);
 if(matcher.matches()||matcher2.matches())
 {
   isValid = true;
 }
 return isValid;
}
public static boolean iswithin70(String text)
{
 if (text.length()<= 70)
 {
   return true;
 }
 else
 {
   return false;
 }
}
}
```

至此，整个实例介绍完毕，执行后的效果如图 7-7 所示，输入手机号码，编写短信内容后，单击“发送”按钮后即可完成短信发送功能，系统会提示发送成功信息，如图 7-8 所示。

如果短信内容和收信人号码格式不规范，会输出对应的错误提示。在本实例中，使用方法 PendingIntent.getBroadcast()自定义了 PendingIntent 并进行 Broadcast 广播，然后使用 SmsManager.getDefault()预先构建的 SmsManager，再使用方法 sendTextMessage()将有关的数据以参数形式带入，这样即可完成发短信

的任务。



图 7-7　执行效果



图 7-8　发送成功

7.5　实现屏幕触控拨号功能

实例 095	屏幕触控拨号程序
源码路径	光盘:\daima\095
视频路径	光盘:\视频\095
实例必备	095.AIDL Service 服务.pdf ① 创建.aidl 文件 ② 实现接口 ③ 向客户端暴露接口 ④ 调用 IPC 方法

7.5.1　实例说明

在触摸屏手机应用中，通常需要触摸一个按钮来实现拨号处理。本实例将使用 Intent 方式把电话号码传递给内置的拨号程序，然后由内置拨号程序实现拨号处理操作。利用方法 startActivity()将程序焦点传送给内置的拨号程序，这样，原来的 Activity 会成为失焦状态，并且还会发生 onPause 暂停事件，直到关闭拨号程序，焦点也交还给原来的 Activity。在具体实现上，先插入了一个按钮，当单击按钮后会调用手机内置的默认拨号界面。

7.5.2　具体实现

编写文件 example.java，当用户单击图标按钮后通过 android.intent.action.CALL_BUTTON 调用默认的拨号界面，主要代码如下所示。

```
private ImageButton myImageButton;
@Override
public void onCreate(Bundle savedInstanceState)
{
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myImageButton = (ImageButton) findViewById(R.id.myImageButton);
        myImageButton.setOnClickListener(new ImageButton.OnClickListener()
        {
          public void onClick(View v)
          {
            /*调用拨号界面*/
            Intent myIntentDial = new Intent("android.intent.action.CALL_BUTTON");
            startActivity(myIntentDial);
          }
        });
    }
}
```

执行后会在屏幕中显示一个图标按钮，如图 7-9 所示。单击按钮后，会自动来到系统内置的默认拨号界面，如图 7-10 所示。



图 7-9　初始效果

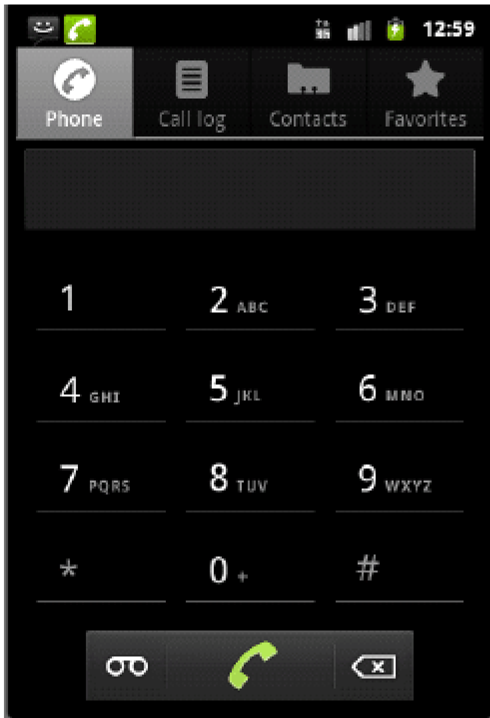


图 7-10　内置的拨号界面

7.6　短信群发系统

实例 096	实现短信群发功能
源码路径	光盘:\daima\096
视频路径	光盘:\视频\096
实例必备	096.BroadcastReceiver 详解.pdf ① BroadcastReceiver 基础 ② Receiver 的生命周期 ③ 基本操作

7.6.1　实例说明

在本实例中，当单击“发送”按钮后会先获取手机通讯录的信息，让用户选择短信接收者。选好

后返回主程序，然后实现短信群发功能。在使用本实例前，需要事先在通讯录中添加一些联系人信息，这些联系人作为接收短信者。当用户选择接收者后，程序会将短信发送给接收者。

7.6.2 具体实现

编写文件 example.java，其具体流程如下所示。

（1）分别定义变量 mTextView01、mTextView3、mTextView5、mButton01 和 mButton02，并为上述变量赋值，主要代码如下所示。

```
private TextView mTextView01;
private TextView mTextView3;
private TextView mTextView5;
private Button mButton01;
private Button mButton02;
/*先声明 strMessage 为 String*/
String strMessage;
private static final int PICK_CONTACT_SUBACTIVITY = 2;
@Override
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   mTextView01 = (TextView)findViewById(R.id.myTextView1);
   mButton01 = (Button)findViewById(R.id.myButton1);
   mTextView3 = (TextView)findViewById(R.id.myTextView3);
   mButton02 = (Button)findViewById(R.id.myButton2);
   mTextView5= (TextView)findViewById(R.id.myTextView5);
```

（2）分别为屏幕中的两个 Button 控件设置处理事件，单击 mButton01 按钮后获取 mTextView3 里的内容，单击 mButton02 按钮后获取 mTextView5 中的内容，主要代码如下所示。

```
   /*设置第一个 Button 事件*/
   mButton01.setOnClickListener(new Button.OnClickListener()
   {
      @Override
      public void onClick(View v)
      {
         Uri uri = Uri.parse("content://contacts/people");
         Intent intent = new Intent(Intent.ACTION_PICK, uri);
         /*获取 mTextView3 中的内容*/
         strMessage = mTextView3.getText().toString();
         startActivityForResult(intent, PICK_CONTACT_SUBACTIVITY);
      }
   });
   /*设置第二个 Button 事件*/
   mButton02.setOnClickListener(new Button.OnClickListener()
   {
      @Override
      public void onClick(View v)
      {
         // TODO Auto-generated method stub
```

```
                Uri uri = Uri.parse("content://contacts/people");
                Intent intent = new Intent(Intent.ACTION_PICK, uri);
                /*获取 mTextView5 中的内容*/
                strMessage = mTextView5.getText().toString();
                startActivityForResult(intent, PICK_CONTACT_SUBACTIVITY);
            }
        });
    }
```

（3）在获取 android.permission.READ_CONTACTS 的权限下，通过 try 来抓取通讯录中存储的姓名、电话号码，然后设置想要发送至的号码，接着用 smsManager.sendTextMessage 发送短信，最后用 Toast 显示正在传送的提示，主要代码如下所示。

```
@Override
protected void onActivityResult
(int requestCode, int resultCode, Intent data)
{
    // TODO Auto-generated method stub
    switch (requestCode)
    {
        case PICK_CONTACT_SUBACTIVITY:
            final Uri uriRet = data.getData();
            if(uriRet != null)
            {
                try
                {
                    /*必须要有 android.permission.READ_CONTACTS 权限*/
                    Cursor c = managedQuery(uriRet, null, null, null, null);
                    c.moveToFirst();
                    /*抓取通讯录中的姓名*/
                    String strName =
                    c.getString(c.getColumnIndexOrThrow(People.NAME));
                    /*抓取通讯录中的电话*/
                    String strPhone =
                    c.getString(c.getColumnIndexOrThrow(People.NUMBER));

                    /*设置要发送至的号码*/
                    String strDestAddress = strPhone;
                    System.out.println(strMessage);
                    SmsManager smsManager = SmsManager.getDefault();

                    PendingIntent mPI = PendingIntent.getBroadcast
                    (example.this, 0, new Intent(), 0);
                    /*发出短信*/
                    smsManager.sendTextMessage
                    (
                        strDestAddress, null, strMessage, mPI, null
                    );
                    /*用 Toast 显示传送中提示*/
                    Toast.makeText
                    (
                        example.this,
```

```
                getString(R.string.str_msg)+strName,
                Toast.LENGTH_SHORT
              ).show();

            mTextView01.setText(strName+":"+strPhone);
          }
          catch(Exception e)
          {
            mTextView01.setText(e.toString());
            e.printStackTrace();
          }
        }
        break;
    }
    super.onActivityResult(requestCode, resultCode, data);
  }
}
```

最后，需要在文件 AndroidManifest.xml 中声明 READ_CONTACTS 权限和 SEND_SMS 权限，主要代码如下所示。

```
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
<uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
```

执行后的效果如图 7-11 所示，单击“发送”按钮后来到联系人界面，如图 7-12 所示。



图 7-11　执行效果



图 7-12　联系人界面

单击其中的一个联系人后，将会发送短信，并输出发送提示，如图 7-13 所示。

上述实例虽然实现了短信发送功能，但是还不能算是群组发送。实际上，Android 系统允许用户创建若干个群组，可以把联系人信息轻松地存储在不同的群组中。Android 系统之所以提供了这样的一个特性，是为了让用户可以给整组联系人群发邮件或者短信，是一个很人性化的功能，如图 7-14 所示。

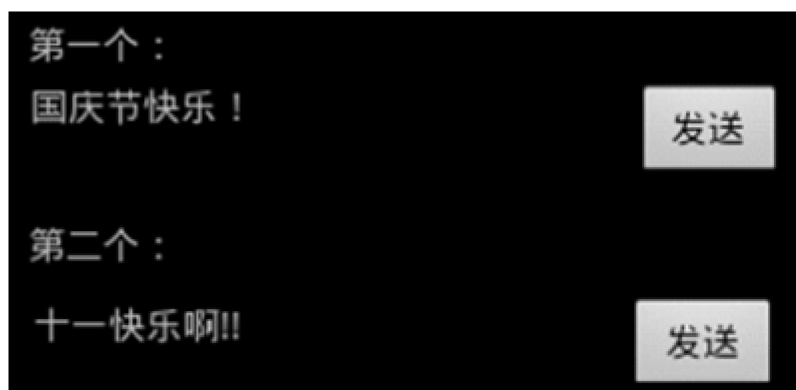


图 7-13　已发送

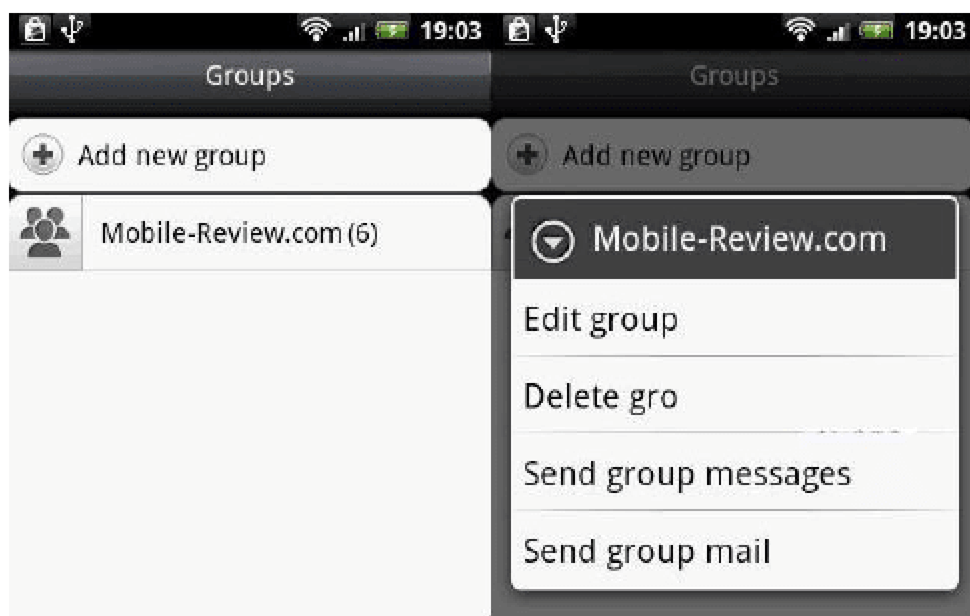


图 7-14　Android 的群组

可以用编程的方式实现群发短信。既可以把联系人的数据用字符串数组的形式保存，也可以以

cursor 为对象，通过循环的方式，在获取联系人数据的同时就传出指定的信息内容。

7.7　监听短信是否发送成功

实例 097	监听短信是否发送成功
源码路径	光盘:\daima\097
视频路径	光盘:\视频\097
实例必备	097.短信处理和电话处理.pdf ① SmsManager 类介绍 ② TelephonyManager 类介绍

7.7.1　实例说明

当发送短信后，用户往往比较关心是否发送成功。在本实例中，当发送一条短信后会及时提供一条说明短信是否发送成功的信息。手机的默认程序可以捕捉到发送状态，这是因为经过系统广播的信息，程序可以捕捉到发送结果。本实例的学习重点是如何衍生广播类 mServiceReceiver，并在这个 Receiver 中判断短信的发送结果。

7.7.2　具体实现

编写文件 example.java，其具体实现流程如下所示。

（1）创建两个 mServiceReceiver 对象作为类成员变量，然后分别创建 mButton1、mTextView01、mEditText1 和 mEditText2 对象，主要代码如下所示。

```
/*创建两个 mServiceReceiver 对象，作为类成员变量*/
private mServiceReceiver mReceiver01, mReceiver02;
private Button mButton1;
private TextView mTextView01;
private EditText mEditText1, mEditText2;
```

（2）自定义 ACTION 常数作为广播的 Intent Filter 识别常数。分别通过 mEditText1 获取电话号码，通过 mEditText2 获取信息内容，然后设置默认为 5556，表示第二个模拟器的 Port，主要代码如下所示。

```
/*自定义 ACTION 常数，作为广播的 Intent Filter 识别常数*/
private String SMS_SEND_ACTIOIN = "SMS_SEND_ACTIOIN";
private String SMS_DELIVERED_ACTION = "SMS_DELIVERED_ACTION";

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mTextView01 = (TextView)findViewById(R.id.myTextView1);
```

```
/*电话号码*/
mEditText1 = (EditText) findViewById(R.id.myEditText1);

/*短信内容*/
mEditText2 = (EditText) findViewById(R.id.myEditText2);
mButton1 = (Button) findViewById(R.id.myButton1);

//mEditText1.setText("+12345678");
/*设置默认为 5556，表示第二个模拟器的 Port*/
mEditText1.setText("5556");
mEditText2.setText("Hello AAA!");
```

（3）设置单击按钮后的事件处理程序，strDestAddress 对象是欲发送的电话号码，strMessage 对象是要发送的内容，主要代码如下所示。

```
/*发送 SMS 短信按钮事件处理*/
mButton1.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View arg0)
  {
    /*欲发送的电话号码*/
    String strDestAddress = mEditText1.getText().toString();
    /*欲发送的短信内容*/
    String strMessage = mEditText2.getText().toString();
```

（4）创建 SmsManager 对象 smsManager 来发送短信，具体流程如下所示。

① 创建自定义 Action 常数的 Intent。

② 用 sentIntent 参数为传送后接收的广播信息 PendingIntent。

③ 用 deliveryIntent 参数为送达后接收的广播信息 PendingIntent。

④ 发送 SMS 短信。

⑤ 若有异常，则用 mTextView01.setText(e.toString())输出异常。

主要代码如下所示。

```
    /*创建 SmsManager 对象*/
    SmsManager smsManager = SmsManager.getDefault();

    // TODO Auto-generated method stub
    try
    {
      /*创建自定义 Action 常数的 Intent（给 PendingIntent 参数之用）*/
      Intent itSend = new Intent(SMS_SEND_ACTIOIN);
      Intent itDeliver = new Intent(SMS_DELIVERED_ACTION);

      /*sentIntent 参数为传送后接收的广播信息 PendingIntent*/
      PendingIntent mSendPI = PendingIntent.getBroadcast
      (getApplicationContext(), 0, itSend, 0);

      /*deliveryIntent 参数为送达后接收的广播信息 PendingIntent*/
      PendingIntent mDeliverPI = PendingIntent.getBroadcast
      (getApplicationContext(), 0, itDeliver, 0);
```

```
            /*发送 SMS 短信，注意倒数的两个 PendingIntent 参数*/
            smsManager.sendTextMessage
            (strDestAddress, null, strMessage, mSendPI, mDeliverPI);

            mTextView01.setText(R.string.str_sms_sending);
          }
          catch(Exception e)
          {
            mTextView01.setText(e.toString());
            e.printStackTrace();
          }
        }
      });
    }
```

（5）自定义 mServiceReceiver 来覆盖 BroadcastReceiver 以聆听短信状态信息。如果发送短信成功，则输出“成功发送”提示，如果发送短信失败，则输出“发送失败”提示，主要代码如下所示。

```
/*自定义 mServiceReceiver 覆盖 BroadcastReceiver 聆听短信状态信息*/
public class mServiceReceiver extends BroadcastReceiver
{
  @Override
  public void onReceive(Context context, Intent intent)
  {
    // TODO Auto-generated method stub

    try
    {
      /*android.content.BroadcastReceiver.getResultCode()方法*/
      switch(getResultCode())
      {
        case Activity.RESULT_OK:
          /*发送短信成功*/
          //mTextView01.setText(R.string.str_sms_sent_success);
          mMakeTextToast
          (
            getResources().getText
            (R.string.str_sms_sent_success).toString(),
            true
          );
          break;
        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
          /*发送短信失败*/
          //mTextView01.setText(R.string.str_sms_sent_failed);
          mMakeTextToast
          (
            getResources().getText
            (R.string.str_sms_sent_failed).toString(),
            true
          );
          break;
```

```
                case SmsManager.RESULT_ERROR_RADIO_OFF:
                    break;
                case SmsManager.RESULT_ERROR_NULL_PDU:
                    break;
            }
        }
        catch(Exception e)
        {
            mTextView01.setText(e.toString());
            e.getStackTrace();
        }
    }
}
```

（6）定义方法 mMakeTextToast()输出发送成功或失败的提示，主要代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)
{
    if(isLong==true)
    {
        Toast.makeText(example.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
        Toast.makeText(example12.this, str, Toast.LENGTH_SHORT).show();
    }
}
```

（7）定义方法 onResume()重启 Activity，主要代码如下所示。

```
@Override
protected void onResume()
{
    /*自定义 IntentFilter 为 SENT_SMS_ACTIOIN Receiver*/
    IntentFilter mFilter01;
    mFilter01 = new IntentFilter(SMS_SEND_ACTIOIN);
    mReceiver01 = new mServiceReceiver();
    registerReceiver(mReceiver01, mFilter01);

    /*自定义 IntentFilter 为 DELIVERED_SMS_ACTION Receiver*/
    mFilter01 = new IntentFilter(SMS_DELIVERED_ACTION);
    mReceiver02 = new mServiceReceiver();
    registerReceiver(mReceiver02, mFilter01);

    super.onResume();
}
```

（8）定义方法 onPause()暂停 Activity，主要代码如下所示。

```
@Override
protected void onPause()
{
    // TODO Auto-generated method stub

    /*取消注册自定义 Receiver*/
    unregisterReceiver(mReceiver01);
```

```
        unregisterReceiver(mReceiver02);

        super.onPause();
    }
}
```

至此，整个实例介绍完毕，发送短信后将显示短信是否发送成功的提示，如图 7-15 所示。



图 7-15　短信提示

第 8 章　二维/三维图形、渲染和动画实战

图形和图像永远是多媒体的重要组成部分，在计算机领域，可以将图形图像分为 2D 和 3D 两大类。本章将通过具体实例的实现流程，详细讲解在 Android 系统中实现 2D 和 3D 图形图像处理的基本方法及实现图片渲染和简单动画效果的方法。

8.1　在手机屏幕中绘制一个矩形

实例 098	在屏幕中绘制一个矩形
源码路径	光盘:\daima\098
视频路径	光盘:\视频\098
实例必备	098.SurfaceFlinger 渲染管理器.pdf ① SurfaceFlinger 基础 ② Surface 和 Canvas

8.1.1　实例说明

在 Android 系统中，可以使用类 Color 和类 Paint 实现绘图功能。其中，Android.Graphics.Color 提供了常规主要颜色的定义，如 Color.BLACK 和 Color.GREEN 等，平时创建时主要使用以下静态方法来实现。

- ☑ static int argb(int alpha, int red, int green, int blue)：构造一个包含透明对象的颜色。
- ☑ static int rgb(int red, int green, int blue)：构造一个标准的颜色对象。
- ☑ static int parseColor(String colorString)：解析一种颜色字符串的值，如传入 Color.BLACK。

本类返回的均为一个整型值，类似于绿色为 0xff00ff00，红色为 0xffff0000。可以将这个 DWORD（双字节）型看做 AARRGGBB 格式的组合，AA 代表 Alpha 透明色，每个分割后的 WORD（单字节）取值范围是 0～255（颜色值范围）。

可以将类 Android.Graphics.Paint 中的 Paint 理解为对画笔、画刷的属性定义。

8.1.2　具体实现

（1）编写文件 Activity.java，通过 mGameView = new GameView(this)，用 Activity 类的 setContentView()方法设置要显示的具体 View 类。文件 Activity.java 的主要代码如下所示。

```
public class Activity01 extends Activity
{
		private GameView mGameView;
```

```
    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        mGameView = new GameView(this);

        setContentView(mGameView);
    }
}
```

（2）编写文件 draw.java，用于绘制指定的图形，此文件的实现流程如下所示。

① 声明 Paint 对象 mPaint，定义方法 draw()，分别用于构建对象和开启线程，具体代码如下所示。

```
    /*声明 Paint 对象*/
    private Paint mPaint  = null;
    public draw(Context context)
    {
        super(context);
        /*构建对象*/
        mPaint = new Paint();
        /*开启线程*/
        new Thread(this).start();
    }
```

② 定义方法 onDraw()，先设置 Paint 格式和颜色，并根据提取的颜色、尺寸、风格、字体和属性实现绘制处理，具体实现代码如下所示。

```
    public void onDraw(Canvas canvas)
    {
        super.onDraw(canvas);
        /*设置 Paint 为无锯齿*/
        mPaint.setAntiAlias(true);
        /*设置 Paint 的颜色*/
        mPaint.setColor(Color.WHITE);
        mPaint.setColor(Color.BLUE);
        mPaint.setColor(Color.YELLOW);
        mPaint.setColor(Color.GREEN);
        /*同样是设置颜色*/
        mPaint.setColor(Color.rgb(255, 0, 0));
        /*提取颜色*/
        Color.red(0xcccccc);
        Color.green(0xcccccc);
        /*设置 Paint 的颜色和 Alpha 值(a,r,g,b)*/
        mPaint.setARGB(255, 255, 0, 0);
        /*设置 Paint 的 Alpha 值*/
        mPaint.setAlpha(220);
        /*这里可以设置为另外一个 Paint 对象*/
        // mPaint.set(new Paint());
        /*设置字体的尺寸*/
        mPaint.setTextSize(14);
        //设置 Paint 的风格为“空心”
        //也可以设置为“实心”（Paint.Style.FILL）
```

```
			mPaint.setStyle(Paint.Style.STROKE);
			//设置“空心”的外框的宽度
			mPaint.setStrokeWidth(5);
			/*得到 Paint 的一些属性*/
			Log.i(TAG, "paint 的颜色：" + mPaint.getColor());
			Log.i(TAG, "paint 的 Alpha：" + mPaint.getAlpha());
			Log.i(TAG, "paint 的外框的宽度：" + mPaint.getStrokeWidth());
			Log.i(TAG, "paint 的字体尺寸：" + mPaint.getTextSize());
			/*绘制一个矩形*/
			//肯定是一个空心的矩形
			canvas.drawRect((320 - 80) / 2, 20, (320 - 80) / 2 + 80, 20 + 40, mPaint);
			/*设置风格为实心*/
			mPaint.setStyle(Paint.Style.FILL);
			mPaint.setColor(Color.GREEN);
			/*绘制绿色实心矩形*/
			canvas.drawRect(0, 20, 40, 20 + 40, mPaint);
		}
```

③ 定义触笔事件 onTouchEvent，设置按键按下时触发事件 onKeyDown，按键弹起触发事件 onKeyUp，主要代码如下所示。

```
		//触笔事件
		public boolean onTouchEvent(MotionEvent event)
		{
			return true;
		}
		//按键按下事件
		public boolean onKeyDown(int keyCode, KeyEvent event)
		{
			return true;
		}
		//按键弹起事件
		public boolean onKeyUp(int keyCode, KeyEvent event)
		{
			return false;
		}
		public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event)
		{
			return true;
		}
		public void run()
		{
			while (!Thread.currentThread().isInterrupted())
			{
				try
				{
					Thread.sleep(100);
				}
				catch (InterruptedException e)
				{
					Thread.currentThread().interrupt();
				}
				//使用 postInvalidate 可以直接在线程中更新界面
```

```
                postInvalidate();
            }
        }
}
```

执行后的效果如图 8-1 所示。

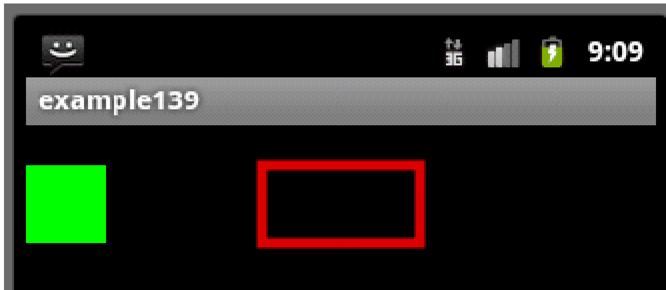


图 8-1　执行效果

8.2　绘制一个画布

实例 099	在手机屏幕中绘制一个画布
源码路径	光盘:\daima\099
视频路径	光盘:\视频\099
实例必备	099.Surface 渲染详解.pdf ① 渲染类 Surface 详解 ② 分析 Layer 和 LayerBuffer

8.2.1　实例说明

在 Android 系统中，使用类 Canvas 实现画布功能。可以将画布看做是一种处理过程，使用各种方法来管理 Bitmap、GL 或者 Path 路径，同时可以配合 Matrix 矩阵类给图像做旋转、缩放等操作，同时 Canvas 类还提供了裁剪、选取等操作。

8.2.2　具体实现

编写主程序文件，主要代码如下所示。

```
    /*声明 Paint 对象*/
    private Paint mPaint = null;
    public example2(Context context)
    {
        super(context);
        /*构建对象*/
        mPaint = new Paint();
        /*开启线程*/
        new Thread(this).start();
    }

    public void onDraw(Canvas canvas)
    {
```

```
        super.onDraw(canvas);

        /*设置画布的颜色*/
        canvas.drawColor(Color.BLACK);

        /*设置取消锯齿效果*/
        mPaint.setAntiAlias(true);

        /*设置裁剪区域*/
        canvas.clipRect(10, 10, 280, 260);

        /*锁定画布*/
        canvas.save();
        /*旋转画布*/
        canvas.rotate(45.0f);

        /*设置颜色及绘制矩形*/
        mPaint.setColor(Color.RED);
        canvas.drawRect(new Rect(15,15,140,70), mPaint);

        /*解除画布的锁定*/
        canvas.restore();

        /*设置颜色及绘制另一个矩形*/
        mPaint.setColor(Color.GREEN);
        canvas.drawRect(new Rect(150,75,260,120), mPaint);
    }

    //触笔事件
    public boolean onTouchEvent(MotionEvent event)
    {
        return true;
    }
    //按键按下事件
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        return true;
    }
    //按键弹起事件
    public boolean onKeyUp(int keyCode, KeyEvent event)
    {
        return false;
    }
    public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event)
    {
        return true;
    }
    public void run()
    {
        while (!Thread.currentThread().isInterrupted())
        {
            try
```

```
                    {
                        Thread.sleep(100);
                    }
                    catch (InterruptedException e)
                    {
                        Thread.currentThread().interrupt();
                    }
                    //使用 postInvalidate 可以直接在线程中更新界面
                    postInvalidate();
                }
            }
        }
```

执行后的效果如图 8-2 所示。

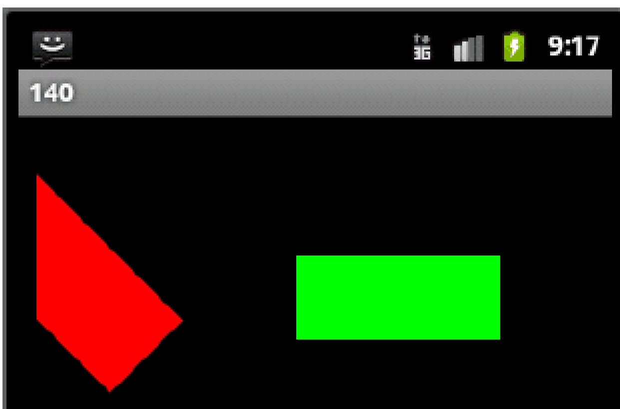


图 8-2　执行效果

8.3　绘制基本的二维图形

实例 100	在手机屏幕中绘制各种二维图形
源码路径	光盘:\daima\100
视频路径	光盘:\视频\100
实例必备	100.Skia 渲染引擎详解.pdf ① Skia 基础 ② Android 中的 Skia

8.3.1　实例说明

在本实例中，将使用 Rect 类来绘制各种样式的图形。Rect 类即 Android.Graphics.Rect 类，也就是矩形区域。类 Rect 除了表示一个矩形区域位置描述外，还可以帮助计算图形之间是否碰撞（包含），对于 Android 游戏开发比较有用，在其主要成员 contains 中包含了如下 3 种重载方法来判断包含关系。

```
boolean contains(int left, int top, int right, int bottom)
boolean contains(int x, int y)
boolean contains(Rect r)
```

8.3.2　具体实现

编写主程序文件，其主要代码如下所示。

```
/*声明 Paint 对象*/
private Paint mPaint = null;
        private example3_1 mGameView2 = null;
public example(Context context)
{
        super(context);
        /*构建对象*/
        mPaint = new Paint();

        mGameView2 = new example3_1(context);

        /*开启线程 */
        new Thread(this).start();
}
public void onDraw(Canvas canvas)
{
        super.onDraw(canvas);

        /*设置画布为黑色背景*/
        canvas.drawColor(Color.BLACK);
        /*取消锯齿*/
        mPaint.setAntiAlias(true);
        mPaint.setStyle(Paint.Style.STROKE);
        {
                /*定义矩形对象*/
                Rect rect1 = new Rect();
                /*设置矩形大小*/
                rect1.left = 5;
                rect1.top = 5;
                rect1.bottom = 25;
                rect1.right = 45;
                mPaint.setColor(Color.BLUE);
                /*绘制矩形*/
                canvas.drawRect(rect1, mPaint);

                mPaint.setColor(Color.RED);
                /*绘制矩形*/
                canvas.drawRect(50, 5, 90, 25, mPaint);
                mPaint.setColor(Color.YELLOW);
                /*绘制圆形（圆心 x，圆心 y，半径 r，p）*/
                canvas.drawCircle(40, 70, 30, mPaint);
                /*定义椭圆对象*/
                RectF rectf1 = new RectF();
                /*设置椭圆大小*/
                rectf1.left = 80;
                rectf1.top = 30;
                rectf1.right = 120;
                rectf1.bottom = 70;
                mPaint.setColor(Color.LTGRAY);
                /*绘制椭圆*/
                canvas.drawOval(rectf1, mPaint);
```

```
        /*绘制多边形*/
        Path path1 = new Path();

        /*设置多边形的点*/
        path1.moveTo(150+5, 80-50);
        path1.lineTo(150+45, 80-50);
        path1.lineTo(150+30, 120-50);
        path1.lineTo(150+20, 120-50);
        /*使这些点构成封闭的多边形*/
        path1.close();
        mPaint.setColor(Color.GRAY);
        /*绘制这个多边形*/
        canvas.drawPath(path1, mPaint);

        mPaint.setColor(Color.RED);
        mPaint.setStrokeWidth(3);
        /*绘制直线*/
        canvas.drawLine(5, 110, 315, 110, mPaint);
    }
    //绘制实心几何体
    mPaint.setStyle(Paint.Style.FILL);
    {
        /*定义矩形对象*/
        Rect rect1 = new Rect();
        /*设置矩形大小*/
        rect1.left = 5;
        rect1.top = 130+5;
        rect1.bottom = 130+25;
        rect1.right = 45;
        mPaint.setColor(Color.BLUE);
        /*绘制矩形*/
        canvas.drawRect(rect1, mPaint);
        mPaint.setColor(Color.RED);
        /*绘制矩形*/
        canvas.drawRect(50, 130+5, 90, 130+25, mPaint);
        mPaint.setColor(Color.YELLOW);
        /*绘制圆形（圆心 x，圆心 y，半径 r，p）*/
        canvas.drawCircle(40, 130+70, 30, mPaint);
        /*定义椭圆对象*/
        RectF rectf1 = new RectF();
        /*设置椭圆大小*/
        rectf1.left = 80;
        rectf1.top = 130+30;
        rectf1.right = 120;
        rectf1.bottom = 130+70;
        mPaint.setColor(Color.LTGRAY);
        /*绘制椭圆*/
        canvas.drawOval(rectf1, mPaint);
        /*绘制多边形*/
        Path path1 = new Path();
        /*设置多边形的点*/
        path1.moveTo(150+5, 130+80-50);
```

```
                path1.lineTo(150+45, 130+80-50);
                path1.lineTo(150+30, 130+120-50);
                path1.lineTo(150+20, 130+120-50);
                /*使这些点构成封闭的多边形*/
                path1.close();
                mPaint.setColor(Color.GRAY);
                /*绘制这个多边形*/
                canvas.drawPath(path1, mPaint);
                mPaint.setColor(Color.RED);
                mPaint.setStrokeWidth(3);
                /*绘制直线*/
                canvas.drawLine(5, 130+110, 315, 130+110, mPaint);
            }
            /*通过 ShapeDrawable 绘制几何图形*/
            mGameView2.DrawShape(canvas);
        }
        //触笔事件
        public boolean onTouchEvent(MotionEvent event)
        {
            return true;
        }
        //按键按下事件
        public boolean onKeyDown(int keyCode, KeyEvent event)
        {
            return true;
        }
        //按键弹起事件
        public boolean onKeyUp(int keyCode, KeyEvent event)
        {
            return false;
        }
        public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event)
        {
            return true;
        }
        public void run()
        {
            while (!Thread.currentThread().isInterrupted())
            {
                try
                {
                    Thread.sleep(100);
                }
                catch (InterruptedException e)
                {
                    Thread.currentThread().interrupt();
                }
                //使用 postInvalidate 可以直接在线程中更新界面
                postInvalidate();
            }
        }
}
```

执行后的效果如图 8-3 所示。

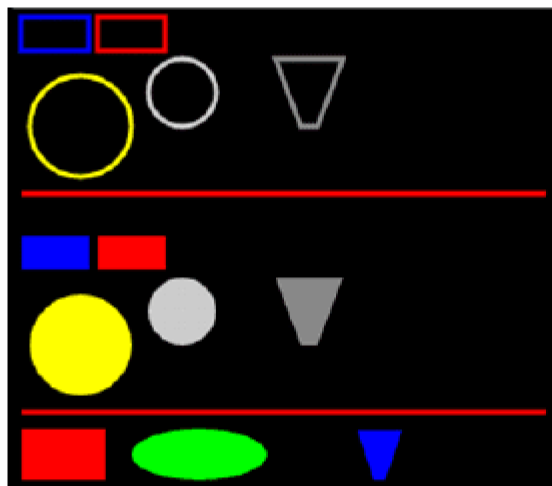


图 8-3　执行效果

8.4　渲染一个几何图形

实例 101	在手机屏幕中渲染一个几何图形
源码路径	光盘:\daima\101
视频路径	光盘:\视频\101
实例必备	101.使用 Skia 绘图.pdf

8.4.1　实例说明

在 Android 系统中，可以使用 Shader 类来渲染图像以及一些几何图形。在使用 Shader 类时需要先构建 Shader 对象，然后通过 Paint 的 setShader()方法设置渲染对象，再设置渲染对象，最后在绘制时使用这个 Paint 对象即可。当然，用不同的渲染时需要构建不同的对象。

8.4.2　具体实现

编写主程序文件，其具体代码如下所示。

```
/*声明 Bitmap 对象*/
Bitmap mBitQQ  = null;
Int BitQQwidth = 0;
Int BitQQheight = 0;
Paint mPaint = null;
/*Bitmap 渲染*/
Shader mBitmapShader = null;
/*线性渐变渲染*/
Shader mLinearGradient = null;
/*混合渲染*/
Shader mComposeShader = null;
/*唤醒渐变渲染*/
Shader mRadialGradient = null;
/*梯度渲染*/
Shader mSweepGradient = null;
ShapeDrawable mShapeDrawableQQ = null;
```

```
    public example5(Context context)
    {
        super(context);

        /*装载资源*/
        mBitQQ = ((BitmapDrawable) getResources().getDrawable(R.drawable.qq)).getBitmap();
        /*得到图片的宽度和高度*/
        BitQQwidth = mBitQQ.getWidth();
        BitQQheight = mBitQQ.getHeight();
        /*创建 BitmapShader 对象*/
        mBitmapShader = new BitmapShader(mBitQQ,Shader.TileMode.REPEAT,Shader.TileMode.MIRROR);
        /*创建 LinearGradient 并设置渐变的颜色数组*/
        mLinearGradient = new LinearGradient(0,0,100,100, new int[ ]{Color.RED,Color.GREEN, Color.BLUE,
Color.WHITE}, null,Shader.TileMode.REPEAT);
        /*混合渲染*/
        mComposeShader = new ComposeShader(mBitmapShader,mLinearGradient,PorterDuff. Mode.
DARKEN);
        /*构建 RadialGradient 对象，设置半径的属性*/
        //这里使用了 BitmapShader 和 LinearGradient 进行混合
        //当然也可以使用其他组合
        //混合渲染的模式很多，可以根据自己的需要来选择
        mRadialGradient = new RadialGradient(50,200,50,new int[ ]{Color.GREEN,Color.RED,Color.BLUE,
Color.WHITE},null,Shader.TileMode.REPEAT);
        /*构建 SweepGradient 对象*/
        mSweepGradient = new SweepGradient(30,30,new int[ ]{Color.GREEN,Color.RED,Color.BLUE, Color.
WHITE},null);
        mPaint = new Paint();

        /*开启线程*/
        new Thread(this).start();
    }
    public void onDraw(Canvas canvas)
    {
        super.onDraw(canvas);

        //将图片裁剪为椭圆形
        /*构建 ShapeDrawable 对象并定义形状为椭圆*/
        mShapeDrawableQQ = new ShapeDrawable(new OvalShape());
        /*设置要绘制的椭圆形的东西为 ShapeDrawable*/
        mShapeDrawableQQ.getPaint().setShader(mBitmapShader);
        /*设置显示区域*/
        mShapeDrawableQQ.setBounds(0,0, BitQQwidth, BitQQheight);
        /*绘制 ShapeDrawableQQ*/
        mShapeDrawableQQ.draw(canvas);
        //绘制渐变的矩形
        mPaint.setShader(mLinearGradient);
        canvas.drawRect(BitQQwidth, 0, 320, 156, mPaint);
        //显示混合渲染效果
        mPaint.setShader(mComposeShader);
        canvas.drawRect(0, 300, BitQQwidth, 300+BitQQheight, mPaint);
        //绘制环形渐变
        mPaint.setShader(mRadialGradient);
        canvas.drawCircle(50, 200, 50, mPaint);
```

```
        //绘制梯度渐变
        mPaint.setShader(mSweepGradient);
        canvas.drawRect(150, 160, 300, 300, mPaint);
    }
    //触笔事件
    public boolean onTouchEvent(MotionEvent event)
    {
        return true;
    }
    //按键按下事件
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        return true;
    }
    //按键弹起事件
    public boolean onKeyUp(int keyCode, KeyEvent event)
    {
        return false;
    }
    public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event)
    {
        return true;
    }
    /*线程处理*/
    public void run()
    {
        while (!Thread.currentThread().isInterrupted())
        {
            try
            {
                Thread.sleep(100);
            }
            catch (InterruptedException e)
            {
                Thread.currentThread().interrupt();
            }
            //使用 postInvalidate 可以直接在线程中更新界面
            postInvalidate();
        }
    }
}
```

执行后的效果如图 8-4 所示。

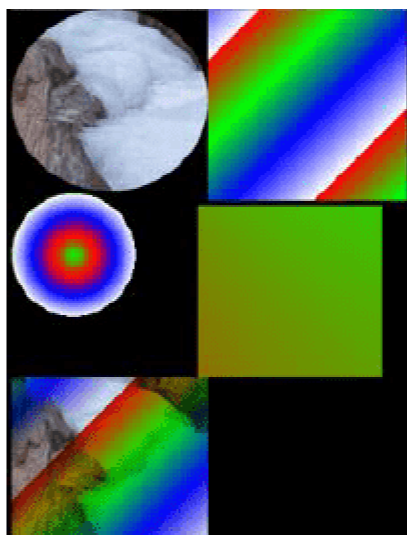


图 8-4　执行效果

8.5 实现动画效果

实例 102	在手机屏幕中实现动画效果
源码路径	光盘:\daima\102
视频路径	光盘:\视频\102
实例必备	102.Skia 的其他功能.pdf

8.5.1 实例说明

在 Android 平台中提供了两类动画，分别是 Tween 动画和 Frame 动画。其中 Tween 动画用于对场景中的对象不断进行图像变换来产生动画效果，而 Frame 动画用于顺序播放事先做好的图像。在本实例中，将使用 Tween 实现动画效果。

8.5.2 具体实现

主程序文件的主要代码如下所示。

```
/*定义 Alpha 动画*/
private Animation mAnimationAlpha = null;

/*定义 Scale 动画*/
private Animation mAnimationScale = null;

/*定义 Translate 动画*/
private Animation mAnimationTranslate = null;

/*定义 Rotate 动画*/
private Animation mAnimationRotate  = null;

/*定义 Bitmap 对象*/
Bitmap mBitQQ  = null;

public example(Context context)
{
	super(context);

	/*装载资源*/
	mBitQQ = ((BitmapDrawable) getResources().getDrawable(R.drawable.qq)).getBitmap();
}
public void onDraw(Canvas canvas)
{
	super.onDraw(canvas);
	/*绘制图片*/
	canvas.drawBitmap(mBitQQ, 0, 0, null);
}
public boolean onKeyUp(int keyCode, KeyEvent event)
```

```
    {
        switch ( keyCode )
        {
        case KeyEvent.KEYCODE_DPAD_UP:
            /*创建 Alpha 动画*/
            mAnimationAlpha = new AlphaAnimation(0.1f, 1.0f);
            /*设置动画的时间*/
            mAnimationAlpha.setDuration(3000);
            /*开始播放动画*/
            this.startAnimation(mAnimationAlpha);
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN:
            /*创建 Scale 动画*/
            mAnimationScale =new ScaleAnimation(0.0f, 1.0f, 0.0f, 1.0f,
            Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
            /*设置动画的时间*/
            mAnimationScale.setDuration(500);
            /*开始播放动画*/
            this.startAnimation(mAnimationScale);
            break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
            /*创建 Translate 动画*/
            mAnimationTranslate = new TranslateAnimation(10, 100,10, 100);
            /*设置动画的时间*/
            mAnimationTranslate.setDuration(1000);
            /*开始播放动画*/
            this.startAnimation(mAnimationTranslate);
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
            /*创建 Rotate 动画*/
            mAnimationRotate=new RotateAnimation(0.0f, +360.0f,
                                    Animation.RELATIVE_TO_SELF,0.5f,
                                    Animation.RELATIVE_TO_SELF, 0.5f);
            /*设置动画的时间*/
            mAnimationRotate.setDuration(1000);
            /*开始播放动画*/
            this.startAnimation(mAnimationRotate);
            break;
        }
        return true;
    }
}
```

执行后的效果如图 8-5 所示。



图 8-5　执行效果

8.6　实现 Frame 动画效果

实例 103	在手机屏幕中实现 Frame 动画效果
源码路径	光盘:\daima\103
视频路径	光盘:\视频\103
实例必备	103.Android 绘图基础.pdf ① 使用 Canvas 画布 ② 使用 Paint 类 ③ 位图操作类 Bitmap

8.6.1　实例说明

Android 动画分为 Tween 动画和 Frame 动画，Tween 动画主要包括图片的放大、缩小、旋转、透明度变化、移动等操作；Frame 动画就是把一张张的图片连续播放产生动画效果。在本实例中，将演示实现 Tween 动画效果的方法。

8.6.2　具体实现

主程序文件的主要代码如下所示。

```
/*定义 AnimationDrawable 动画*/
private AnimationDrawable frameAnimation = null;
Context mContext = null;

/*定义一个 Drawable 对象*/
Drawable mBitAnimation = null;
public example10(Context context)
{
	super(context);

	mContext = context;

	/*实例化 AnimationDrawable 对象*/
	frameAnimation = new AnimationDrawable();

	/*装载资源*/
	//这里用一个循环装载所有名字类似的资源
	//这个方法用处非常大
	for (int i = 1; i <= 15; i++)
	{
		int id = getResources().getIdentifier("a" + i, "drawable", mContext.getPackageName());
		mBitAnimation = getResources().getDrawable(id);
		/*为动画添加一帧*/
```

```
                //参数 mBitAnimation 是该帧的图片
                //参数 500 是该帧显示的时间，按毫秒计算
                frameAnimation.addFrame(mBitAnimation, 500);
            }

            /*设置播放模式是否循环，false 表示循环，true 表示不循环*/
            frameAnimation.setOneShot( false );

            /*设置本类将要显示这个动画*/
            this.setBackgroundDrawable(frameAnimation);
        }

        public void onDraw(Canvas canvas)
        {
            super.onDraw(canvas);

        }

        public boolean onKeyUp(int keyCode, KeyEvent event)
        {
            switch ( keyCode )
            {
            case KeyEvent.KEYCODE_DPAD_UP:
                /*开始播放动画*/
                frameAnimation.start();
                break;
            }
            return true;
        }
}
```

执行后可以通过按键盘上的方向键来实现动画效果，如图 8-6 所示。



图 8-6　执行效果

8.7　旋转屏图片

实例 104	旋转屏幕中的图片
源码路径	光盘:\daima\104
视频路径	光盘:\视频\104
实例必备	104.使用设置文本颜色类 Color.pdf

8.7.1 实例说明

Matrix 可以操作图像，可以分为 translate（平移）、rotate（旋转）、scale（缩放）和 skew（倾斜）4 种操作方式，每一种操作变换在 Android 的 API 中都提供了 set、post 和 pre 3 种操作方式，除了 translate，其他 3 种操作都可以指定中心点。在本实例中，通过 Bitmap 和 Matrix 实现了对图片的旋转处理。

8.7.2 具体实现

编写旋转处理文件 example.java，其具体实现流程如下所示。

（1）加载默认的 Drawable。

（2）实现向左旋转按钮处理 mButton1.setOnClickListener。

（3）实现向右旋转按钮处理 mButton2.setOnClickListener。

主程序文件的主要实现代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   mButton1 =(Button) findViewById(R.id.myButton1);
   mButton2 =(Button) findViewById(R.id.myButton2);
   mTextView1 = (TextView) findViewById(R.id.myTextView1);
   mImageView1 = (ImageView) findViewById(R.id.myImageView1);
   ScaleTimes = 1;
   ScaleAngle = 1;
   final Bitmap mySourceBmp =
   BitmapFactory.decodeResource(getResources(), R.drawable.hippo);
   final int widthOrig = mySourceBmp.getWidth();
   final int heightOrig = mySourceBmp.getHeight();
   /*程序刚运行，加载默认的 Drawable*/
   mImageView1.setImageBitmap(mySourceBmp);
   /*向左旋转按钮*/
   mButton1.setOnClickListener(new Button.OnClickListener()
   {
      @Override
      public void onClick(View v)
      {
         // TODO Auto-generated method stub
         ScaleAngle--;
         if(ScaleAngle<-5)
         {
            ScaleAngle = -5;
         }
         /*ScaleTimes=1 表示维持 1：1 的宽高比例*/
         int newWidth = widthOrig * ScaleTimes;
         int newHeight = heightOrig * ScaleTimes;
```

```
        float scaleWidth = ((float) newWidth) / widthOrig;
        float scaleHeight = ((float) newHeight) / heightOrig;

        Matrix matrix = new Matrix();
        /*使用 matrix.postScale 设置维度*/
        matrix.postScale(scaleWidth, scaleHeight);

        /*使用 matrix.postRotate 旋转 Bitmap*/
        matrix.setRotate(5*ScaleAngle);

        /*创建新的 Bitmap 对象*/
        Bitmap resizedBitmap =
        Bitmap.createBitmap
        (mySourceBmp, 0, 0, widthOrig, heightOrig, matrix, true);
        BitmapDrawable myNewBitmapDrawable =
        new BitmapDrawable(resizedBitmap);

        mImageView1.setImageDrawable(myNewBitmapDrawable);
        mTextView1.setText(Integer.toString(5*ScaleAngle));
    }
});

/*向右旋转按钮*/
mButton2.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        ScaleAngle++;
        if(ScaleAngle>5)
        {
            ScaleAngle = 5;
        }

        /*ScaleTimes=1，维持 1：1 的宽高比例*/
        int newWidth = widthOrig * ScaleTimes;
        int newHeight = heightOrig * ScaleTimes;

        /*计算旋转的 Matrix 比例*/
        float scaleWidth = ((float) newWidth) / widthOrig;
        float scaleHeight = ((float) newHeight) / heightOrig;

        Matrix matrix = new Matrix();
        /*使用 Matrix.postScale 设置维度*/
        matrix.postScale(scaleWidth, scaleHeight);

        /*使用 Matrix.postRotate 方法旋转 Bitmap*/
        //matrix.postRotate(5*ScaleAngle);
        matrix.setRotate(5*ScaleAngle);

        /*创建新的 Bitmap 对象*/
```

```
            Bitmap resizedBitmap =
            Bitmap.createBitmap
            (mySourceBmp, 0, 0, widthOrig, heightOrig, matrix, true);

            BitmapDrawable myNewBitmapDrawable =
            new BitmapDrawable(resizedBitmap);

            mImageView1.setImageDrawable(myNewBitmapDrawable);
            mTextView1.setText(Integer.toString(5*ScaleAngle));
        }
      });
    }
}
```

执行后将显示一幅图片和两个按钮，分别单击“左转”和“右转”按钮后会实现对图片旋转处理，分别如图 8-7 和图 8-8 所示。



图 8-7　左旋转效果

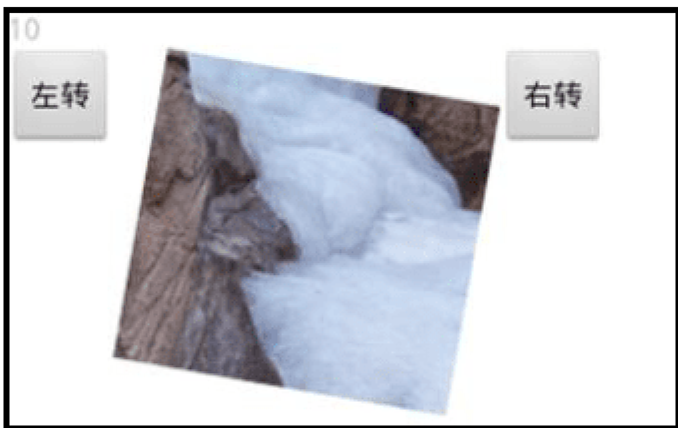


图 8-8　右旋转后的效果

8.8　实现满天星动画效果

实例 105	在屏幕中实现天上移动星星的效果
源码路径	光盘:\daima\105
视频路径	光盘:\视频\105
实例必备	105.使用矩形类 Rect 和 RectF.pdf ① 类 Rect ② 类 RectF

8.8.1　实例说明

本实例需要实现三维效果，所以需要使用 OpenGL（Open Graphics Library，开放的图形程序）技术。在实现本实例时，需要事先准备一幅素材图像，然后在此图像的基础上形成一个螺旋图案，以达到闪烁星星的效果。为了实现旋转效果，定义变量 angle 表示当前星星所处的角度。

8.8.2　具体实现

编写主程序文件，在此文件中通过创建循环的方式实现星星闪烁的效果，具体实现代码如下所示。

```
        public static final int  num = 50;                          //星星数目
        boolean twinkle = true;                                     //闪烁的星星
        boolean key;
        public Star[ ] star = new Star[num];                        //存放星星的数组
        float zoom= -10.0f;                                         //星星离观察者的距离
        float tilt = 90.0f;                                         //星星的倾角
        float spin;                                                 //闪烁星星的自转
        int one = 0x10000;
        Random random = new Random();
        Int texture;                                                //纹理

        IntBuffer coord = IntBuffer.wrap(new int[ ]{
                0,0,
                one,0,
                one,one,
                0,one,
        });
        IntBuffer vertexs = IntBuffer.wrap(new int[ ]{
                -one,-one,0,
                one,-one,0,
                one,one,0,
                -one,one,0,
        });
        ByteBuffer indices = ByteBuffer.wrap(new byte[ ]{
                1, 0, 2, 3
        });
        @Override
        public void onDrawFrame(GL10 gl)
        {
gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);//清除屏幕和深度缓存
            gl.glBindTexture(GL10.GL_TEXTURE_2D, texture);  //选择纹理
            for (int i=0; i<num; i++)                             //循环设置所有的星星
            {
                gl.glLoadIdentity();                              //绘制每颗星星之前，重置模型观察矩阵
                gl.glTranslatef(0.0f,0.0f,zoom);                  //深入屏幕里面
                gl.glRotatef(tilt,1.0f,0.0f,0.0f);                //倾斜视角
                gl.glRotatef(star[i].angle,0.0f,1.0f,0.0f);       //旋转至当前所画星星的角度
                gl.glTranslatef(star[i].dist,0.0f,0.0f);          //沿 x 轴正向移动
                gl.glRotatef(-star[i].angle,0.0f,1.0f,0.0f);      //取消当前星星的角度
                gl.glRotatef(-tilt,1.0f,0.0f,0.0f);               //取消屏幕倾斜
                //设置定点数组
                gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
                //设置颜色数组
                gl.glEnableClientState(GL10.GL_COLOR_ARRAY);

                gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);

                if (twinkle)                                      //启用闪烁效果
                {
                    //使用 byte 型数值指定一个颜色
                    gl.glColor4f((float)star[(num-i)-1].r/255.0f,(float)star[(num-i)-1].g/255.0f,(float)star[(num-i)-1].
```

```
b/255.0f,1.0f);
                    gl.glVertexPointer(3, GL10.GL_FIXED, 0, vertexs);
                    gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, coord);

                    {
                         coord.position(0);
                         vertexs.position(0);
                         indices.position(0);

                         gl.glDrawElements(GL10.GL_TRIANGLE_STRIP, 4, GL10.GL_UNSIGNED_BYTE, indices);
                    }
                    //绘制结束
                    gl.glFinish();
               }
               gl.glRotatef(spin,0.0f,0.0f,1.0f);                  //绕 z 轴旋转星星
               //使用 byte 型数值指定一个颜色
              gl.glColor4f((float)star[(num-i)-1].r/255.0f,  (float)star[(num-i)-1].g/255.0f,  (float)star[(num-i)-1].b/255.0f,
1.0f);
               gl.glVertexPointer(3, GL10.GL_FIXED, 0, vertexs);
               gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, coord);

               {
                    coord.position(0);
                    vertexs.position(0);
                    indices.position(0);

                    gl.glDrawElements(GL10.GL_TRIANGLE_STRIP, 4, GL10.GL_UNSIGNED_BYTE, indices);
               }
              //绘制正方形结束
              gl.glFinish();
              gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
              //取消顶点数组
              gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
              gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);

               spin+=0.01f;                                        //星星的自转
               star[i].angle+=(float)(i)/(float)num;               //改变星星的自转角度
               star[i].dist-=0.01f;                                //改变星星离中心的距离
               if (star[i].dist<0.0f)                              //星星是否到达中心
               {
                    star[i].dist+=5.0f;                            //往外移 5 个单位
                    star[i].r=random.nextInt(256);                 //赋一个新红色分量
                    star[i].g=random.nextInt(256);                 //赋一个新绿色分量
                    star[i].b=random.nextInt(256);                 //赋一个新蓝色分量
               }
          }
     }
     @Override
     public void onSurfaceChanged(GL10 gl, int width, int height)
     {
          float ratio = (float) width / height;
```

```
        //设置 OpenGL 场景的大小
        gl.glViewport(0, 0, width, height);
        //设置投影矩阵
        gl.glMatrixMode(GL10.GL_PROJECTION);
        //重置投影矩阵
        gl.glLoadIdentity();
        //设置视口的大小
        gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);
        //选择模型观察矩阵
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        //重置模型观察矩阵
        gl.glLoadIdentity();
    }
    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config)
    {
        gl.glShadeModel(GL10.GL_SMOOTH);                //启用阴影平滑
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);        //黑色背景
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_NICEST);
        //告诉系统对透视进行修正
        IntBuffer buffer = IntBuffer.allocate(1);
        //创建一个纹理
        gl.glGenTextures(1, buffer);
        texture = buffer.get();
        //创建一个线性滤波纹理
        gl.glBindTexture(GL10.GL_TEXTURE_2D, texture);
        gl.glTexParameterx(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_LINEAR);
        gl.glTexParameterx(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);
        GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, GLImage.mBitmap, 0);
        gl.glEnable(GL10.GL_TEXTURE_2D);                //启用纹理映射
        gl.glShadeModel(GL10.GL_SMOOTH);                //启用阴影平滑
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);        //黑色背景
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_NICEST);//真正精细的透视修正
        gl.glBlendFunc(GL10.GL_SRC_ALPHA,GL10.GL_ONE);  //设置混色函数取得半透明效果
        gl.glEnable(GL10.GL_BLEND);                     //启用混色
        for (int i=0; i<num; i++)                       //创建循环设置全部星星
        {
            Star starTMP = new Star();
            starTMP.angle=0.0f;                         //所有星星都从零角度开始
            starTMP.dist=((float)(i)/(float)num)*5.0f;  //计算星星离中心的距离
            starTMP.r=random.nextInt(256);              //为 star[loop]设置随机红色分量
            starTMP.g=random.nextInt(256);              //为 star[loop]设置随机绿色分量
            starTMP.b=random.nextInt(256);              //为 star[loop]设置随机蓝色分量
            star[i] = starTMP;
        }
    }
    public boolean onKeyUp(int keyCode, KeyEvent event)
    {
        twinkle=!twinkle;
        return false;
    }
```

```
}
class Star
{
	Int r, g, b;                          //星星的颜色
	float dist;                           //星星距离中心的距离
	float angle = 0.0f;                   //当前星星所处的角度
}
```

执行后的效果如图 8-9 所示。

在本实例中用到了 OpenGL 技术，OpenGL 是一个定义了跨编程语言、跨平台的编程接口的规格，用于三维图像（二维的亦可）。OpenGL 是一个专业的图形程序接口，也是一个功能强大，调用方便的底层图形库。OpenGL 的前身是 SGI 公司为其图形工作站开发的 IRIS GL。IRIS GL 是一个工业标准的 3D 图形软件接口，功能虽然强大但是移植性不好，于是 SGI 公司便在 IRIS GL 的基础上开发了 OpenGL。虽然 DirectX 在家用市场全面领先，但在专业高端绘图领域，OpenGL 是不能被取代的主角。



图 8-9　执行效果

8.9　构建一个模拟 3D 场景

实例 106	在屏幕中构建一个模拟 3D 场景效果
源码路径	光盘:\daima\106
视频路径	光盘:\视频\106
实例必备	106.非矢量图形拉伸类 NinePatch.pdf

8.9.1　实例说明

在现实世界中，任何一个复杂的对象都是由一些简单的形状构成的。所以在创建复杂环境之前应该先定义一个场景的数据结构。例如，可以使用 3D 空间中的坐标值（x,y,z）以及纹理坐标（u,v）来定义一个三角形的顶点。

8.9.2　具体实现

编写文件 Data1.java 来设置顶点结构 VERTEX 和三角形结构 TRIANGLE，绘制 3D 场景，并定义方法 SetupWorld()用于读取资源数据，去掉每个三角形的顶点数据，定义方法 onKeyUp()来响应键盘按键事件，具体实现代码如下所示。

```
//VERTEX 顶点结构
class VERTEX
{
	float x, y, z;              //3D 坐标
	float u, v;                 //纹理坐标
	public VERTEX(float x,float y,float z,float u,float v)
	{
```

```
        this.x = x;
        this.y = y;
        this.z = z;
        this.u = u;
        this.v = v;
    }
}
//TRIANGLE 三角形结构
class TRIANGLE
{
    //VERTEX 矢量数组，大小为 3
    VERTEX[ ]vertex = new VERTEX[3];
}
//SECTOR 区段结构
class SECTOR
{
    //Sector 中的三角形个数
    int numtriangles;
    //三角行的 list
    List<TRIANGLE>    triangle = new ArrayList<TRIANGLE>();
}
        //开始绘制 3D 场景
        for(TRIANGLE triangle : sector1.triangle)
        {
         vertexPointer.clear();
         texCoordPointer.clear();
         gl.glNormal3f(0.0f, 0.0f, 1.0f);
         for(int i=0; i<3; i++)
         {
             VERTEX vt = triangle.vertex[i];
             vertexPointer.put(vt.x);
             vertexPointer.put(vt.y);
             vertexPointer.put(vt.z);
             texCoordPointer.put(vt.u);
             texCoordPointer.put(vt.v);
         }
         gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 4);
        }
        gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
    //读取资源数据
    public void SetupWorld()
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(GLFile.getFile("data/world.txt")));

        TRIANGLE triangle = new TRIANGLE();
        int vertexIndex = 0;

        try {
            String line = null;
            while((line = br.readLine()) != null){
                if(line.trim().length() <= 0 || line.startsWith("/")){
```

```
                    continue;
                }
                String part[ ] = line.trim().split("\\s+");
                float x = Float.valueOf(part[0]);
                float y = Float.valueOf(part[1]);
                float z = Float.valueOf(part[2]);
                float u = Float.valueOf(part[3]);
                float v = Float.valueOf(part[4]);
                VERTEX vertex = new VERTEX(x, y, z, u, v);
                triangle.vertex[vertexIndex] = vertex;

                vertexIndex ++;
                if(vertexIndex == 3){
                    vertexIndex = 0;
                    sector1.triangle.add(triangle);
                    triangle = new TRIANGLE();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public boolean onKeyUp(int keyCode, KeyEvent event)
    {
        switch ( keyCode )
        {
            case KeyEvent.KEYCODE_DPAD_LEFT:
                yrot -= 1.5f;
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                yrot += 1.5f;
                break;
            case KeyEvent.KEYCODE_DPAD_UP:
                //沿游戏者所在的 x 平面移动
                xpos -= (float)Math.sin(heading*piover180) * 0.05f;
                //沿游戏者所在的 z 平面移动
                zpos -= (float)Math.cos(heading*piover180) * 0.05f;
                if (walkbiasangle >= 359.0f)      //如果 walkbiasangle 大于 359°
                {
                    walkbiasangle = 0.0f;       //将 walkbiasangle 设为 0
                }
                else
                {
                     walkbiasangle+= 10;        //如果 walkbiasangle < 359° ，则增加 10
                }
                //使游戏者产生跳跃感
                walkbias = (float)Math.sin(walkbiasangle * piover180)/20.0f;
                break;
            case KeyEvent.KEYCODE_DPAD_DOWN:
                //沿游戏者所在的 x 平面移动
                xpos += (float)Math.sin(heading*piover180) * 0.05f;
                //沿游戏者所在的 z 平面移动
                zpos += (float)Math.cos(heading*piover180) * 0.05f;
```

```
                    //如果 walkbiasangle 小于 1°
                    if (walkbiasangle <= 1.0f)
                    {
                         walkbiasangle = 359.0f;    //使 walkbiasangle 等于 359
                    }
                    else
                    {
                         walkbiasangle-= 10;          //如果 walkbiasangle > 1，则减去 10
                    }
                    //使游戏者产生跳跃感
                    walkbias = (float)Math.sin(walkbiasangle * piover180)/20.0f;
                    break;
          }
          return false;
     }
```

执行后的效果如图 8-10 所示。



图 8-10　执行效果

8.10　实现粒子系统效果

实例 107	在手机屏幕中模拟实现粒子系统效果
源码路径	光盘:\daima\107
视频路径	光盘:\视频\107
实例必备	107.使用变换处理类 Matrix.pdf

8.10.1　实例说明

粒子系统表示三维计算机图形学中模拟一些特定的模糊现象的技术，而这些现象用其他传统的渲染技术难以实现真实感的 Game Physics（游戏物理世界）。经常使用粒子系统模拟的现象有火、爆炸、烟、水流、火花、落叶、云、雾、雪、尘、流星尾迹或者像发光轨迹这样的抽象视觉效果等。在本实例中，使用了三角形绘制一个粒子效果，把某一个“点”抽象成一个物体，然后赋予这个物体一些特效属性。

8.10.2　具体实现

（1）在布局文件中设置插入一个 TextView。

（2）定义类 lizi 来表示“点”，在其中定义了各个点的坐标变量，具体代码如下所示。

```
public class lizi
{
        boolean active;
        float life;
        float fade;
        float r;
        float g;
        float b;
        float x;
        float y;
        float z;
        float xi;
        float yi;
        float zi;
        float xg;
        float yg;
        float zg;
}
```

（3）为了更好地操作和控制微粒，在文件 example152.java 中特意加入了如下变量。

```
    public final static int MAX_PARTICLES =1000;
    boolean rainbow=true;
    Random random = new Random();
    float slowdown=0.5f;                /*减速粒子*/
    float xspeed=1;                     /*x 方向的速度*/
    float yspeed=3;                     /*y 方向的速度*/
    float zoom=-30.0f;                  /*沿 z 轴缩放*/
    int loop;                           /*循环变量*/
    int col=0;                          /*当前的颜色*/
    int delay;                          /*延迟彩虹效果*/
```

（4）定义数组 colors，用于存储 12 种不同的颜色，具体代码如下所示。

```
    static float colors[ ][ ]=
    {
        {1.0f,  0.5f,  0.5f},
        {1.0f,  0.75f, 0.5f},
        {1.0f,  1.0f,  0.5f},
        {0.75f, 1.0f,  0.5f},
        {0.5f,  1.0f,  0.5f},
        {0.5f,  1.0f,  0.75f},
        {0.5f,  1.0f,  1.0f},
        {0.5f,  0.75f, 1.0f},
        {0.5f,  0.5f,  1.0f},
        {0.75f, 0.5f,  1.0f},
        {1.0f,  0.5f,  1.0f},
        {1.0f,  0.5f,  0.75f}
    };
```

（5）装载纹理贴图来实现初始化处理，具体代码如下所示。

```
    public void ResetParticle(int num, int color, float xDir, float yDir, float zDir)
    {
```

```
        particle tmp = new particle();
        tmp.active=true;
        tmp.life=1.0f;
        tmp.fade=(float)(rand()%100)/1000.0f+0.003f;
        tmp.r=colors[color][0];
        tmp.g=colors[color][1];
        tmp.b=colors[color][2];
        tmp.x=0.0f;
        tmp.y=0.0f;
        tmp.z=0.0f;
        tmp.xi=xDir;
        tmp.yi=yDir;
        tmp.zi=zDir;
        tmp.xg=0.0f;
        tmp.yg=-0.5f;
        tmp.zg=0.0f;
        particles[num] = tmp;
        return;
    }
```

（6）给粒子分配一种颜色，通过方法 onDrawFrame()实现对粒子的绘制处理，具体代码如下所示。

```
    public void onDrawFrame(GL10 gl)
    {
        FloatBuffer vertices = FloatBuffer.wrap(new float[12]);
        FloatBuffer texcoords = FloatBuffer.wrap(new float[8]);
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertices);
        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, texcoords);
        gl.glLoadIdentity();
        for (loop = 0; loop < MAX_PARTICLES; loop++)
        {
            if (particles[loop].active)
            {
                float x = particles[loop].x;
                float y = particles[loop].y;
                float z = particles[loop].z + zoom;
                gl.glColor4f(particles[loop].r, particles[loop].g, particles[loop].b, particles[loop].life);
                texcoords.clear();
                vertices.clear();
                texcoords.put(1.0f);
                texcoords.put(1.0f);
                vertices.put(x + 0.5f);
                vertices.put(y + 0.5f);
                vertices.put(z);
                texcoords.put(0.0f);
                texcoords.put(1.0f);
                vertices.put(x - 0.5f);
```

```
                vertices.put(y + 0.5f);
                vertices.put(z);
                texcoords.put(1.0f);
                texcoords.put(0.0f);
                vertices.put(x + 0.5f);
                vertices.put(y - 0.5f);
                vertices.put(z);
                texcoords.put(0.0f);
                texcoords.put(0.0f);
                vertices.put(x - 0.5f);
                vertices.put(y - 0.5f);
                vertices.put(z);
                gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
                particles[loop].x += particles[loop].xi / (slowdown * 1000);
                particles[loop].y += particles[loop].yi / (slowdown * 1000);
                particles[loop].z += particles[loop].zi / (slowdown * 1000);
                particles[loop].xi += particles[loop].xg;
                particles[loop].yi += particles[loop].yg;
                particles[loop].zi += particles[loop].zg;
                particles[loop].life -= particles[loop].fade;
                if (particles[loop].life < 0.0f)
                {
                    float xi, yi, zi;
                    xi = xspeed + (float) ((rand() % 60) - 32.0f);
                    yi = yspeed + (float) ((rand() % 60) - 30.0f);
                    zi = (float) ((rand() % 60) - 30.0f);
                    ResetParticle(loop, col, xi, yi, zi);
                }
            }
        }
        gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glFinish();
    }
```

执行后的效果如图 8-11 所示。

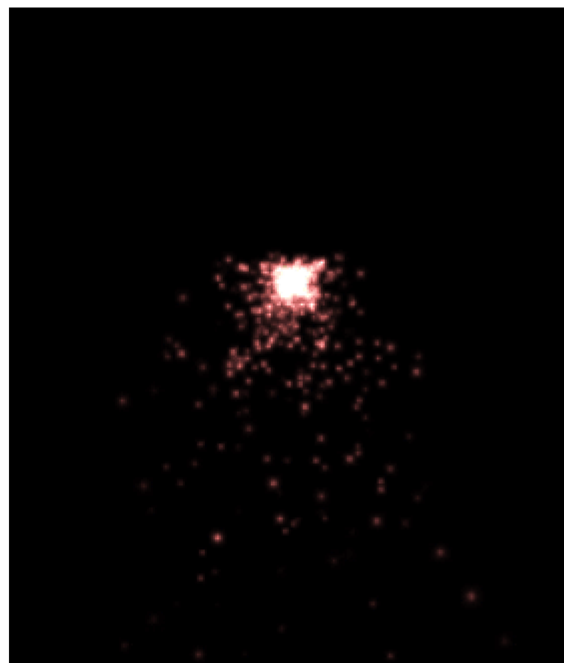


图 8-11　执行效果

8.11　绘制一个三维圆柱体

实例 108	在手机屏幕中绘制一个三维圆柱体
源码路径	光盘:\daima\108
视频路径	光盘:\视频\108
实例必备	108.使用 BitmapFactory 类.pdf

8.11.1　实例说明

绘制圆柱体时需要运用到立体几何中的圆柱体知识，使用圆柱体可以构建 3D 场景中的柱子模型类等事物。在本实例中，使用 OpenGL ES 技术在屏幕中绘制了一个三维效果的圆柱。为了实现逼真的三维效果，通过鼠标可以拖动查看圆柱的不同方位。

8.11.2　具体实现

（1）编写文件 Activity_GL.java，设置界面是可触控的，便于对该界面进行控制，其主要代码如下所示。

```
public class Activity_GL extends Activity {
        private MySurfaceView mGLSurfaceView;
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            requestWindowFeature(Window.FEATURE_NO_TITLE);
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.
LayoutParams.FLAG_FULLSCREEN);
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

            mGLSurfaceView = new MySurfaceView(this);
            setContentView(mGLSurfaceView);
            mGLSurfaceView.setFocusableInTouchMode(true);          //设置为可触控
            mGLSurfaceView.requestFocus();                          //获取焦点
      }
      @Override
      protected void onResume() {
            super.onResume();
            mGLSurfaceView.onResume();
      }
      @Override
      protected void onPause() {
            super.onPause();
            mGLSurfaceView.onPause();
      }
}
```

（2）编写文件 MySurfaceView.java，在此定义了圆柱体场景类 MySurfaceView，此类实现加载和渲染场景的功能。文件 MySurfaceView.java 的主要代码如下所示。

```
public class MySurfaceView extends GLSurfaceView {
    private final float TOUCH_SCALE_FACTOR = 180.0f/320;               //角度缩放比例
    private SceneRenderer mRenderer;                                    //场景渲染器
    private float mPreviousY;                                           //上次的触控位置 y 坐标
    private float mPreviousX;                                           //上次的触控位置 x 坐标
     private int lightAngle=90;                                         //灯的当前角度
     public MySurfaceView(Context context) {
        super(context);
        mRenderer = new SceneRenderer();                                //创建场景渲染器
        setRenderer(mRenderer);                                         //设置渲染器
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);           //设置渲染模式为主动渲染
    }
    //触摸事件回调方法
    @Override
    public boolean onTouchEvent(MotionEvent e) {
        float y = e.getY();
        float x = e.getX();
        switch (e.getAction()) {
        case MotionEvent.ACTION_MOVE:
            float dy = y - mPreviousY;                                  //计算触控笔 y 位移
            float dx = x - mPreviousX;                                  //计算触控笔 x 位移
            mRenderer.cylinder.mAngleX += dy * TOUCH_SCALE_FACTOR;      //设置沿 x 轴旋转角度
            mRenderer.cylinder.mAngleZ += dx * TOUCH_SCALE_FACTOR;      //设置沿 z 轴旋转角度
            requestRender();                                            //重绘画面
        }
        mPreviousY = y;                                                 //记录触控笔位置
        mPreviousX = x;                                                 //记录触控笔位置
        return true;
    }
     private class SceneRenderer implements GLSurfaceView.Renderer
    {
          int textureId;                                                //纹理名称 ID
          DrawY cylinder;                                               //创建圆柱体

     public SceneRenderer()
     {
     }

        public void onDrawFrame(GL10 gl) {
            //清除颜色缓存
            gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
            //设置当前矩阵为模式矩阵
            gl.glMatrixMode(GL10.GL_MODELVIEW);
            //设置当前矩阵为单位矩阵
            gl.glLoadIdentity();

            gl.glPushMatrix();                                          //保护变换矩阵现场
            float lx=0;                                                 //设定光源的位置
```

```
            float ly=(float)(7*Math.cos(Math.toRadians(lightAngle)));
            float lz=(float)(7*Math.sin(Math.toRadians(lightAngle)));
            float[ ] positionParamsRed={lx,ly,lz,0};
            gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_POSITION, positionParamsRed,0);

            initMaterial(gl);                               //初始化纹理
            gl.glTranslatef(0, 0, -10f);                    //平移
            initLight(gl);                                  //开灯
            cylinder.drawSelf(gl);                          //绘制
            closeLight(gl);                                 //关灯

            gl.glPopMatrix();                               //恢复变换矩阵现场
        }
        public void onSurfaceChanged(GL10 gl, int width, int height) {
            //设置视窗大小及位置
            gl.glViewport(0, 0, width, height);
            //设置当前矩阵为投影矩阵
            gl.glMatrixMode(GL10.GL_PROJECTION);
            //设置当前矩阵为单位矩阵
            gl.glLoadIdentity();
            //计算透视投影的比例
            float ratio = (float) width / height;
            //调用此方法计算产生透视投影矩阵
            gl.glFrustumf(-ratio, ratio, -1, 1, 1, 100);
        }
        public void onSurfaceCreated(GL10 gl, EGLConfig config) {
            //关闭抗抖动
            gl.glDisable(GL10.GL_DITHER);
            //设置特定 Hint 项目的模式，这里设置为使用快速模式
            gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
            //设置屏幕背景色为黑色 RGBA
            gl.glClearColor(0,0,0,0);
            //设置着色模型为平滑着色
            gl.glShadeModel(GL10.GL_SMOOTH);
            //启用深度测试
            gl.glEnable(GL10.GL_DEPTH_TEST);

            textureId=initTexture(gl,R.drawable.stone);         //纹理 ID
            cylinder=new DrawY(10f,2f,18f,textureId);           //创建圆柱体

        }
    }
    //初始化白色灯
    private void initLight(GL10 gl)
    {
        gl.glEnable(GL10.GL_LIGHTING);                          //允许光照
        gl.glEnable(GL10.GL_LIGHT1);                            //打开 1 号灯

        //环境光设置
        float[ ] ambientParams={0.2f,0.2f,0.2f,1.0f};           //光参数 RGBA
        gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_AMBIENT, ambientParams,0);
```

```
        //散射光设置
        float[ ] diffuseParams={1f,1f,1f,1.0f};                    //光参数 RGBA
        gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_DIFFUSE, diffuseParams,0);

        //反射光设置
        float[ ] specularParams={1f,1f,1f,1.0f};                   //光参数 RGBA
        gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_SPECULAR, specularParams,0);
    }

    //关闭灯
    private void closeLight(GL10 gl)
    {
         gl.glDisable(GL10.GL_LIGHT1);
         gl.glDisable(GL10.GL_LIGHTING);
    }

    //初始化材质
    private void initMaterial(GL10 gl)
    {
        //环境光
        float ambientMaterial[ ] = {248f/255f, 242f/255f, 144f/255f, 1.0f};
        gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, ambientMaterial,0);
        //散射光
        float diffuseMaterial[ ] = {248f/255f, 242f/255f, 144f/255f, 1.0f};
        gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, diffuseMaterial,0);
        //高光材质
        float specularMaterial[ ] = {248f/255f, 242f/255f, 144f/255f, 1.0f};
        gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_SPECULAR, specularMaterial,0);
        gl.glMaterialf(GL10.GL_FRONT_AND_BACK, GL10.GL_SHININESS, 100.0f);
    }

    //初始化纹理
    public int initTexture(GL10 gl,int drawableId)//textureId
    {
         //生成纹理 ID
         int[ ] textures = new int[1];
         gl.glGenTextures(1, textures, 0);
         int currTextureId=textures[0];
         gl.glBindTexture(GL10.GL_TEXTURE_2D, currTextureId);
         gl.glTexParameterf(GL10.GL_TEXTURE_2D,  GL10.GL_TEXTURE_MIN_FILTER,GL10.GL_LINEAR_
MIPMAP_NEAREST);


        gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MAG_FILTER,  GL10.GL_LINEAR_
MIPMAP_LINEAR);
        ((GL11)gl).glTexParameterf(GL10.GL_TEXTURE_2D,  GL11.GL_GENERATE_MIPMAP,  GL10.GL_
TRUE);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_S,GL10.GL_REPEAT);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_T,GL10.GL_REPEAT);
        InputStream is = this.getResources().openRawResource(drawableId);
```

```
            Bitmap bitmapTmp;
            try
            {
              bitmapTmp = BitmapFactory.decodeStream(is);
            }
            finally
            {
                  try
                  {
                        is.close();
                  }
                  catch(IOException e)
                  {
                        e.printStackTrace();
                  }
            }
            GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmapTmp, 0);
            bitmapTmp.recycle();

            return currTextureId;
      }
}
```

（3）编写文件 DrawY.java，在此实现了圆柱类的三角形绘制方法的构造器部分的内容。在本实例的三角形绘制方法中添加了光照和纹理贴图。文件 DrawY.java 的主要代码如下所示。

```
public class DrawY
{
      private FloatBuffer myVertexBuffer;                    //顶点坐标缓冲
      private FloatBuffer myNormalBuffer;                    //法向量缓冲
      private FloatBuffer myTexture;                         //纹理缓冲

      int textureId;

      int vCount;                                            //顶点数量

      float length;                                          //圆柱长度
      float circle_radius;                                   //圆截环半径
      float degreespan;                                      //圆截环每一份的度数大小

      public float mAngleX;
      public float mAngleY;
      public float mAngleZ;
      public DrawY(float length,float circle_radius,float degreespan,int textureId)
      {
            this.circle_radius=circle_radius;
            this.length=length;
            this.degreespan=degreespan;
            this.textureId=textureId;

            float collength=(float)length;                   //圆柱每块所占的长度
            int spannum=(int)(360.0f/degreespan);

            ArrayList<Float> val=new ArrayList<Float>();     //顶点存放列表
            ArrayList<Float> ial=new ArrayList<Float>();     //法向量存放列表
```

```
for(float circle_degree=180.0f;circle_degree>0.0f;circle_degree-=degreespan)//循环行
{
        float x1 =(float)(-length/2);
        float y1=(float) (circle_radius*Math.sin(Math.toRadians(circle_degree)));
        float z1=(float) (circle_radius*Math.cos(Math.toRadians(circle_degree)));

        float a1=0;
        float b1=y1;
        float c1=z1;
        float l1=getVectorLength(a1, b1, c1);      //模长
        a1=a1/l1;                                  //法向量规格化
        b1=b1/l1;
        c1=c1/l1;

        float x2 =(float)(-length/2);
        float y2=(float) (circle_radius*Math.sin(Math.toRadians(circle_degree-degreespan)));
        float z2=(float) (circle_radius*Math.cos(Math.toRadians(circle_degree-degreespan)));

        float a2=0;
        float b2=y2;
        float c2=z2;
        float l2=getVectorLength(a2, b2, c2);      //模长
        a2=a2/l2;                                  //法向量规格化
        b2=b2/l2;
        c2=c2/l2;

        float x3 =(float)(length/2);
        float y3=(float) (circle_radius*Math.sin(Math.toRadians(circle_degree-degreespan)));
        float z3=(float) (circle_radius*Math.cos(Math.toRadians(circle_degree-degreespan)));

        float a3=0;
        float b3=y3;
        float c3=z3;
        float l3=getVectorLength(a3, b3, c3);      //模长
        a3=a3/l3;                                  //法向量规格化
        b3=b3/l3;
        c3=c3/l3;

        float x4 =(float)(length/2);
        float y4=(float) (circle_radius*Math.sin(Math.toRadians(circle_degree)));
        float z4=(float) (circle_radius*Math.cos(Math.toRadians(circle_degree)));

        float a4=0;
        float b4=y4;
        float c4=z4;
        float l4=getVectorLength(a4, b4, c4);      //模长
        a4=a4/l4;                                  //法向量规格化
        b4=b4/l4;
        c4=c4/l4;

        val.add(x1);val.add(y1);val.add(z1);       //两个三角形，共 6 个顶点的坐标
        val.add(x2);val.add(y2);val.add(z2);
```

```
                val.add(x4);val.add(y4);val.add(z4);

                val.add(x2);val.add(y2);val.add(z2);
                val.add(x3);val.add(y3);val.add(z3);
                val.add(x4);val.add(y4);val.add(z4);

                ial.add(a1);ial.add(b1);ial.add(c1);          //顶点对应的法向量
                ial.add(a2);ial.add(b2);ial.add(c2);
                ial.add(a4);ial.add(b4);ial.add(c4);

                ial.add(a2);ial.add(b2);ial.add(c2);
                ial.add(a3);ial.add(b3);ial.add(c3);
                ial.add(a4);ial.add(b4);ial.add(c4);
        }

        vCount=val.size()/3;                                   //确定顶点数量

        //顶点
        float[ ] vertexs=new float[vCount*3];
        for(int i=0;i<vCount*3;i++)
        {
                vertexs[i]=val.get(i);
        }
        ByteBuffer vbb=ByteBuffer.allocateDirect(vertexs.length*4);
        vbb.order(ByteOrder.nativeOrder());
        myVertexBuffer=vbb.asFloatBuffer();
        myVertexBuffer.put(vertexs);
        myVertexBuffer.position(0);

        //法向量
        float[ ] normals=new float[vCount*3];
        for(int i=0;i<vCount*3;i++)
        {
                normals[i]=ial.get(i);
        }
        ByteBuffer ibb=ByteBuffer.allocateDirect(normals.length*4);
        ibb.order(ByteOrder.nativeOrder());
        myNormalBuffer=ibb.asFloatBuffer();
        myNormalBuffer.put(normals);
        myNormalBuffer.position(0);

        //纹理
        float[ ] textures=generateTexCoor(spannum);
        ByteBuffer tbb=ByteBuffer.allocateDirect(textures.length*4);
        tbb.order(ByteOrder.nativeOrder());
        myTexture=tbb.asFloatBuffer();
        myTexture.put(textures);
        myTexture.position(0);
}
......
//法向量规格化，求模长度
public float getVectorLength(float x,float y,float z)
{
```

```
        float pingfang=x*x+y*y+z*z;
        float length=(float) Math.sqrt(pingfang);
        return length;
    }

    //自动切分纹理产生纹理数组的方法
    public float[ ] generateTexCoor(int bh)
    {
     float[ ] result=new float[bh*6*2];
     float REPEAT=2;
     float sizeh=1.0f/bh;//行数
     int c=0;
     for(int i=0;i<bh;i++)
     {
            //每行列一个矩形，由两个三角形构成，共 6 个点，12 个纹理坐标
            float t=i*sizeh;

            result[c++]=0;
            result[c++]=t;

            result[c++]=0;
            result[c++]=t+sizeh;

            result[c++]=REPEAT;
            result[c++]=t;

            result[c++]=0;
            result[c++]=t+sizeh;

            result[c++]=REPEAT;
            result[c++]=t+sizeh;

            result[c++]=REPEAT;
            result[c++]=t;
     }
     return result;
    }
```

执行后在屏幕中显示一个三维圆柱，如图 8-12 所示。通过鼠标可以查看此圆柱的不同部位，如图 8-13 所示。

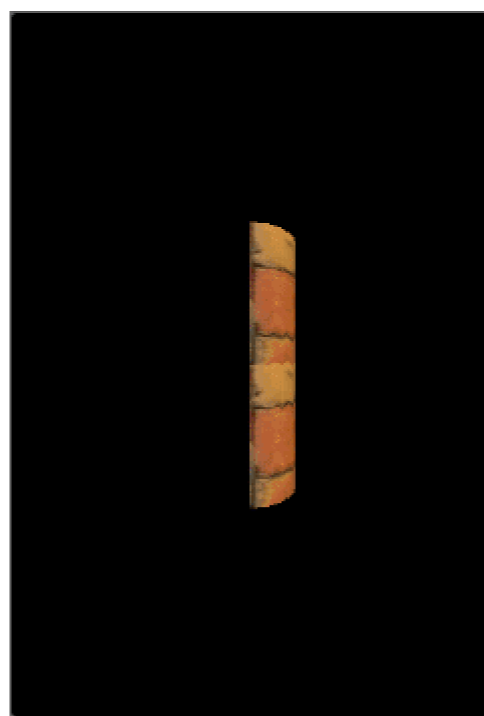


图 8-12　执行效果

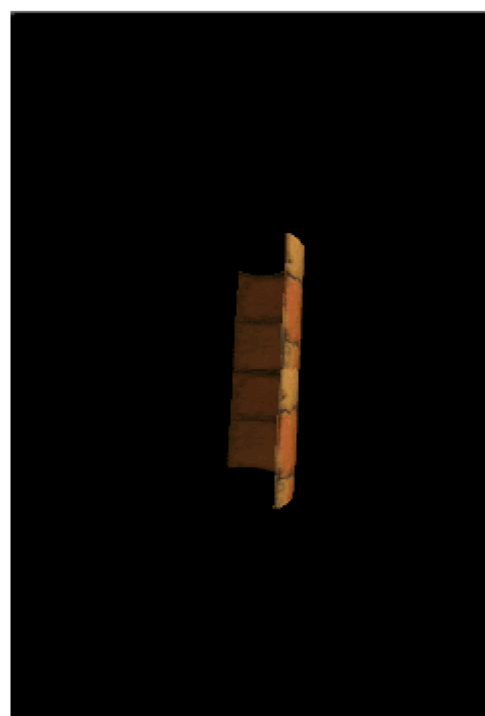


图 8-13　鼠标拖动后的效果

8.12　混 合 图 像

实例 109	在手机屏幕中混合图像
源码路径	光盘:\daima\109
视频路径	光盘:\视频\109
实例必备	109.使用 Region 类.pdf

8.12.1　实例说明

在本实例中，根据源混合因子和目标混合因子的组合原理来混合手机屏幕中的图像。本实例的目的是讲解源混合因子和目标混合因子组合的用法，有两种混合模式的组合，其中一个为参数 GL ONE 和 GL_ONE_MINUS_DST_ALPHA 的混合，另一个为参数 GL SRC_COLOR 和 GL_DST_ALPHA 的混合效果。

8.12.2　具体实现

（1）编写文件 MySurfaceView.java，在此实现了绘制场景类 MySurfaceView，主要代码如下所示。

```
private SceneRenderer mRenderer;                                          //场景渲染器
    public MySurfaceView(Context context) {
        super(context);
        mRenderer = new SceneRenderer();                                  //创建场景渲染器
        setRenderer(mRenderer);                                           //设置渲染器
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);  //设置渲染模式为主动渲染
    }
    private class SceneRenderer implements GLSurfaceView.Renderer
    {
          final int one=65535;
          int baseTextureId;                                        //最底层矩形的不透明纹理的纹理 ID
          int topTextureId;                                         //顶层透明纹理的纹理 ID
          ColorF c1;                                                //颜色矩形 1
          ColorF c2;                                                //颜色矩形 2
          Test t1;                                                  //纹理矩形 1
          Test t2;                                                  //纹理矩形 2
        public void onDrawFrame(GL10 gl) {
            //采用平滑着色
            gl.glShadeModel(GL10.GL_SMOOTH);

            //清除颜色缓存于深度缓存
            gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
            //设置当前矩阵为模式矩阵
            gl.glMatrixMode(GL10.GL_MODELVIEW);
            //设置当前矩阵为单位矩阵
            gl.glLoadIdentity();
```

```
        //绘制底层纹理矩形
        gl.glPushMatrix();
        gl.glTranslatef(0, 0f, -2f);
        t1.drawSelf(gl);
        gl.glPopMatrix();
        //绘制上层纹理矩形
        gl.glPushMatrix();
        gl.glTranslatef(-0.7f, -0.3f, -1.9f);
        t2.drawSelf(gl);
        gl.glPopMatrix();
        //绘制上层颜色半透明矩形
        gl.glPushMatrix();
        gl.glTranslatef(0.7f, 0.4f, -1.8f);
        c1.drawSelf(gl);
        gl.glPopMatrix();

        //绘制上层颜色半透明矩形
        gl.glPushMatrix();
        gl.glTranslatef(-0.6f, 0.6f, -1.8f);
        c2.drawSelf(gl);
        gl.glPopMatrix();
    }
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        //设置视窗大小及位置
        gl.glViewport(0, 0, width, height);
        //设置当前矩阵为投影矩阵
        gl.glMatrixMode(GL10.GL_PROJECTION);
        //设置当前矩阵为单位矩阵
        gl.glLoadIdentity();
        //计算透视投影的比例
        float ratio = (float) width / height;
        //调用此方法计算产生透视投影矩阵
        gl.glFrustumf(-ratio, ratio, -1, 1, 1, 100);
    }
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        //关闭抗抖动
        gl.glDisable(GL10.GL_DITHER);
        //设置特定 Hint 项目的模式，这里设置为使用快速模式
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
        //设置屏幕背景色黑色 RGBA
        gl.glClearColor(0,0,0,0);
        //启用深度测试
        gl.glEnable(GL10.GL_DEPTH_TEST);
        //开启混合
        gl.glEnable(GL10.GL_BLEND);
        gl.glBlendFunc(GL10.GL_SRC_COLOR, GL10.GL_DST_ALPHA);

        //初始化纹理
        baseTextureId=initTexture(gl,R.drawable.base);
        topTextureId=initTexture(gl,R.drawable.top);

        //创建矩形
```

```
            c1=new ColorF(one,0,0,one*3/4);
            c2=new ColorF(0,one,0,one/2);
            t1=new Test(baseTextureId);
            t2=new Test(topTextureId);
        }
    }

    //初始化纹理
    public int initTexture(GL10 gl,int drawableId)//textureId
    {
        //生成纹理 ID
        int[ ] textures = new int[1];
        gl.glGenTextures(1, textures, 0);
        int currTextureId=textures[0];
        gl.glBindTexture(GL10.GL_TEXTURE_2D, currTextureId);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_NEAREST);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_ LINEAR);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_S,GL10.GL_CLAMP_TO_ EDGE);
        gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_T,GL10.GL_CLAMP_TO_ EDGE);

       InputStream is = this.getResources().openRawResource(drawableId);
       Bitmap bitmapTmp;
       try
       {
         bitmapTmp = BitmapFactory.decodeStream(is);
       }
       finally
       {
           try
           {
               is.close();
           }
           catch(IOException e)
           {
               e.printStackTrace();
           }
       }
       GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmapTmp, 0);
       bitmapTmp.recycle();
        return currTextureId;
    }
```

（2）编写文件 Test.java，分别实现顶点坐标数据缓冲和顶点着色数据缓冲，根据坐标实现绘制功能，并在绘制完成后关闭纹理功能。文件 Test.java 的主要代码如下所示。

```
public class Test {
    private IntBuffer    mVertexBuffer;                    //顶点坐标数据缓冲
    private FloatBuffer    mTextureBuffer;                 //顶点着色数据缓冲
    int vCount;
    int texId;
    public Test(int texId)
    {
     this.texId=texId;
```

```
        //顶点坐标数据的初始化
        vCount=6;
        final int UNIT_SIZE=40000;
        int vertices[ ]=new int[ ]
        {
          -1*UNIT_SIZE,1*UNIT_SIZE,0,
          -1*UNIT_SIZE,-1*UNIT_SIZE,0,
          1*UNIT_SIZE,1*UNIT_SIZE,0,

          -1*UNIT_SIZE,-1*UNIT_SIZE,0,
          1*UNIT_SIZE,-1*UNIT_SIZE,0,
          1*UNIT_SIZE,1*UNIT_SIZE,0
        };

        //创建顶点坐标数据缓冲
        //vertices.length*4 是因为一个整数为 4 个字节
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
        vbb.order(ByteOrder.nativeOrder());                //设置字节顺序
        mVertexBuffer = vbb.asIntBuffer();                 //转换为 int 型缓冲
        mVertexBuffer.put(vertices);                       //向缓冲区中放入顶点坐标数据
        mVertexBuffer.position(0);                         //设置缓冲区起始位置
        //特别提示：由于不同平台字节顺序不同，数据单元不是字节的一定要经过 ByteBuffer
        //转换，关键是要通过 ByteOrder 设置 nativeOrder()，否则可能出现问题
        float textures[ ]=new float[ ]
        {
          0,0,0,1,1,0,
          0,1,1,1,1,0
        };
        //创建顶点纹理数据缓冲
        ByteBuffer tbb = ByteBuffer.allocateDirect(textures.length*4);
        tbb.order(ByteOrder.nativeOrder());                //设置字节顺序
        mTextureBuffer= tbb.asFloatBuffer();               //转换为 Float 型缓冲
        mTextureBuffer.put(textures);                      //向缓冲区中放入顶点着色数据
        mTextureBuffer.position(0);                        //设置缓冲区起始位置
        //特别提示：由于不同平台字节顺序不同，数据单元不是字节的一定要经过 ByteBuffer
        //转换，关键是要通过 ByteOrder 设置 nativeOrder()，否则可能出现问题
        //顶点纹理数据的初始化
    }
    public void drawSelf(GL10 gl)
    {
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);  //启用顶点坐标数组
        //为画笔指定顶点坐标数据
        gl.glVertexPointer
        (
                3,                                     //每个顶点的坐标数量为 3
                GL10.GL_FIXED,                         //顶点坐标值的类型为 GL_FIXED
                0,                                     //连续顶点坐标数据之间的间隔
                MVertexBuffer                          //顶点坐标数据
        );
        //开启纹理
        gl.glEnable(GL10.GL_TEXTURE_2D);
```

```
            //允许使用纹理 ST 坐标缓冲
            gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
            //为画笔指定纹理 ST 坐标缓冲
            gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
            //绑定当前纹理
            gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);
            //绘制图形
            gl.glDrawArrays
            (
                GL10.GL_TRIANGLES,                    //以三角形方式填充
                0,
                vCount
            );
            //关闭纹理
            gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
            gl.glDisable(GL10.GL_TEXTURE_2D);
      }
}
```

执行后将在屏幕中混合 3 幅指定的图片素材，如图 8-14 所示。

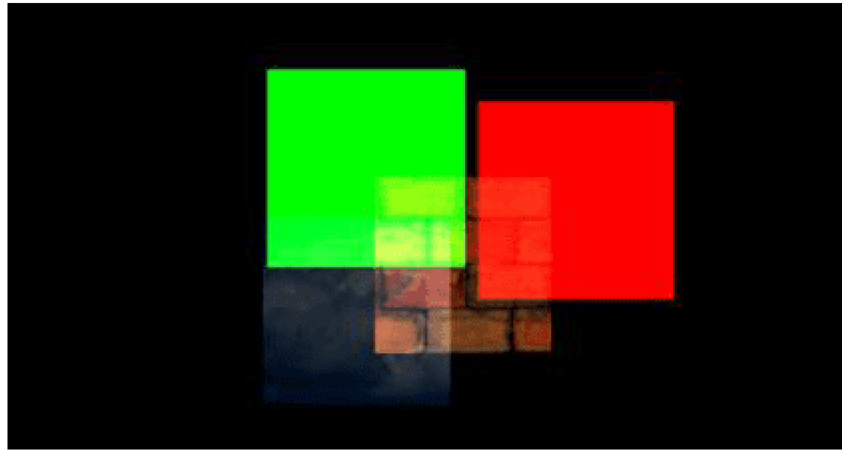


图 8-14　执行效果

第 9 章　网络实战应用

21 世纪的前 10 年被称为信息时代，互联网是信息时代的产物。互联网的推出，直接改变了人们的日常生活。本章将通过几个典型实例的实现过程详细介绍在 Android 系统中与互联网应用相关的基本知识。

9.1　在手机中浏览网页

实例 110	在手机屏幕中浏览网页
源码路径	光盘:\daima\110
视频路径	光盘:\视频\110
实例必备	110.HTTP 基础.pdf ① HTTP 概述 ② HTTP 协议的功能 ③ Android 中的 HTTP

9.1.1　实例说明

在 Android 系统中内置了一个名为 WebKit 的引擎，使用里面的 WebView Widget 可以迅速浏览网页。本实例是通过 WebView.loadUrl 来加载网址的。所以从 EditText 中传入要浏览的网址后，即可在 WebView 中加载网页的内容。

9.1.2　具体实现

编写主程序文件，通过 setOnClickListener 监听按钮单击事件，单击网址后面的箭头后会抓取 EditText 中的数据，然后打开此网址，并在 WebView 中显示网页内容，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   mImageButton1 = (ImageButton)findViewById(R.id.myImageButton1);
   mEditText1 = (EditText)findViewById(R.id.myEditText1);
   mWebView1 = (WebView) findViewById(R.id.myWebView1);

   /*当单击箭头后*/
   mImageButton1.setOnClickListener(new
                                   ImageButton.OnClickListener()
```

```
    {
        @Override
        public void onClick(View arg0)
        {
            // TODO Auto-generated method stub
            {
                mImageButton1.setImageResource(R.drawable.go_2);
                /*抓取 EditText 中的数据*/
                String strURI = (mEditText1.getText().toString());
                /*WebView 显示网页内容*/
                mWebView1.loadUrl(strURI);
                Toast.makeText(
                    example2.this,getString(R.string.load)+strURI,
                                Toast.LENGTH_LONG)
                    .show();
            }
        }
    });
}
```

执行后显示一个文本框，在此可以输入网址，如图 9-1 所示。输入网址并单击后面的▶按钮后，将显示此网页的内容，如图 9-2 所示。



图 9-1　输入网址

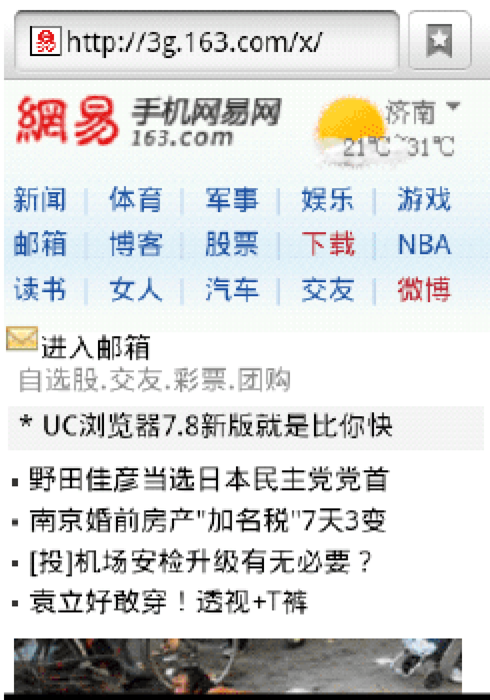


图 9-2　打开的网页

9.2　在手机中加载 HTML 程序

实例 111	在手机中加载 HTML 程序
源码路径	光盘:\daima\111
视频路径	光盘:\视频\111
实例必备	111.HTML 简介.pdf ① HTML 初步 ② 字体格式设置 ③ 使用标识标记

9.2.1 实例说明

HTML 语言是当前主流的网页技术，而 WebView 是一个嵌入式的浏览器，在其中可以直接使用 WebView.loadData()。WebView 将 HTML 标记传递给 WebView 对象，让 Android 手机程序变为 Web 浏览器。这样，网页程序放在 WebView 中运行，如同一个 Web Application。

9.2.2 具体实现

编写主程序文件，在 loadData 中插入预先设置好的 HTML 代码，通过 HTML 代码显示一幅图片和文字，并且实现超链接功能，具体代码如下所示。

```
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
mWebView1 = (WebView) findViewById(R.id.myWebView1);
/*自行设置 WebView 要显示的网页内容*/
mWebView1.
   loadData(
   "<html><body><p>aaaaaaa</p>" +
   "<div class='widget-content'> "+
   "<a href=http://www.sohu.com>" +
   "<img src=http://hiphotos.baidu.com/chaojihedan/pic/item/bbddf5efc260f133fdfa3cd8.jpg />" +
   "<a href=http://www.sohu.com>Link Blog</a>" +
   "</body></html>", "text/html", "utf-8");
```

执行后将显示 HTML 产生的页面，如图 9-3 所示。单击超链接后会跳转到指定的目标页面。

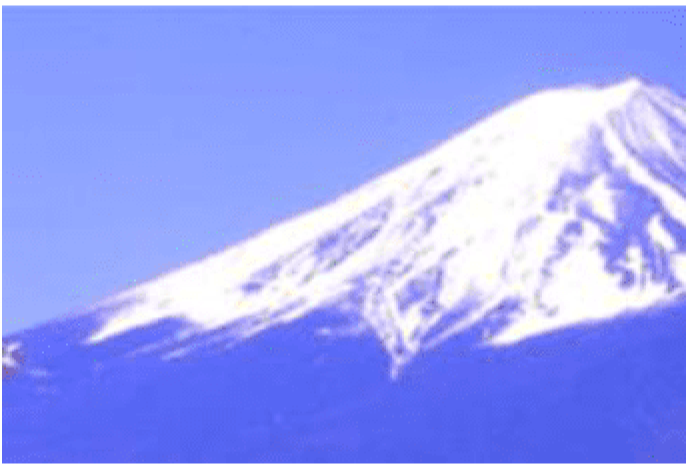


图 9-3 执行效果

9.3 使用内置浏览器打开网页

实例 112	使用内置浏览器打开网页
源码路径	光盘:\daima\112
视频路径	光盘:\视频\112
实例必备	112.使用标准的 Java 接口.pdf ① IP 地址 ② URL 地址 ③ 套接字 socket 类

9.3.1　实例说明

在 Android 系统中有一个内置浏览器，通过这个内置浏览器可以打开网页。在本实例中定义了一个 ListView 控件，其中列表显示了 4 个菜单，单击菜单后会连接到指定的页面。当单击 ListView 中的某一个选项后，会通过 Intent(Intent.ACTION_VIEW,uri)打开内置的浏览器，并浏览 ListView 中创建的网页 URL。

9.3.2　具体实现

编写主程序文件，具体实现流程如下所示。

（1）通过 findViewById 构造器创建 ListView 与 TextView 对象，将 string.xml 中的值信息导入到列表中，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  /*通过 findViewById 构造器创建 ListView 与 TextView 对象*/
  mListView1 =(ListView) findViewById(R.id.myListView1);
  mTextView1 = (TextView) findViewById(R.id.myTextView1);
  mTextView1.setText(getResources().getString(R.string.hello));
  /*将列表通过 string.xml 导入*/
  myFavor = new String[ ] {
                              getResources().getString
                              (R.string.str_list_url1),
                              getResources().getString
                              (R.string.str_list_url2),
                              getResources().getString
                              (R.string.str_list_url3),
                              getResources().getString
                              (R.string.str_list_url4)
                        };
```

（2）将自定义 ArrayAdapter 对象中的信息传入到 ListView 列表中，然后打开 ListAdapter 的可选（Focusable）菜单选项，最后设置单击 ListView 列表选项的处理事件，具体代码如下所示。

```
  /*自定义一个 ArrayAdapter 准备传入 ListView 中，并将 myFavor 列表以参数传入*/
  ArrayAdapter<String> adapter = new
  ArrayAdapter<String>
  (example4.this, android.R.layout.simple_list_item_1, myFavor);
  /*将自定义完成的 ArrayAdapter 传入自定义的 ListView 中*/
  mListView1.setAdapter(adapter);
  /*将 ListAdapter 的可选（Focusable）菜单选项打开*/
  mListView1.setItemsCanFocus(true);
  /*设置 ListView 菜单选项为每次只能单一选项*/
  mListView1.setChoiceMode
  (ListView.CHOICE_MODE_SINGLE);
  /*设置 ListView 选项的 nItemClickListener*/
```

```
mListView1.setOnItemClickListener
(new ListView.OnItemClickListener()
...
```

（3）当用户单击一个 Item 选项后进行比较处理，并从 string.xml 中取出对应的 URL 网址，然后将字符串转换为 URL 对象。具体代码如下所示。

```
/*覆盖 OnItemClick()方法*/
public void onItemClick
(AdapterView<?> arg0, View arg1, int arg2,long arg3)
{
  // TODO Auto-generated method stub
  /*若所选菜单的文字与 myFavor 字符串数组第 1 个文字相同*/
  if(arg0.getAdapter().getItem(arg2).toString()==
  myFavor[0].toString())
  {
    /*取得网址并调用 goToUrl()方法*/
    myUrl=getResources().getString(R.string.str_url1);
    goToUrl(myUrl);
  }
  /*若所选菜单的文字与 myFavor 字符串数组第 2 个文字相同*/
  else if (arg0.getAdapter().getItem(arg2).toString()==
  myFavor[1].toString())
  {
    /*取得网址并调用 goToUrl()方法*/
    myUrl=getResources().getString(R.string.str_url2);
    goToUrl(myUrl);
  }
  /*若所选菜单的文字与 myFavor 字符串数组第 3 个文字相同*/
  else if (arg0.getAdapter().getItem(arg2).toString()==
  myFavor[2].toString())
  {
    /*取得网址并调用 goToUrl()方法*/
    myUrl=getResources().getString(R.string.str_url3);
    goToUrl(myUrl);
  }
  /*若所选菜单的文字与 myFavor 字符串数组第 4 个文字相同*/
  else if (arg0.getAdapter().getItem(arg2).toString()==
  myFavor[3].toString())
  {
    /*取得网址并调用 goToUrl()方法*/
    myUrl=getResources().getString(R.string.str_url4);
    goToUrl(myUrl);
  }
  /*以上皆非*/
  else
  {
    /*显示错误信息*/
    mTextView1.setText("Ooops!!出错了");
  }
}
```

（4）定义方法 goToUrl(String url)打开网址为 URL 的网页，具体代码如下所示。

```
/*打开网页的方法*/
private void goToUrl(String url)
```

```
{
   Uri uri = Uri.parse(url);
   Intent intent = new Intent(Intent.ACTION_VIEW, uri);
   startActivity(intent);
}
```

执行后将列表显示 4 个菜单，如图 9-4 所示。当选择一个菜单后，会显示对应的目标页面。



图 9-4　4 个菜单

9.4　将文件上传至服务器

实例 113	将文件上传至服务器
源码路径	光盘:\daima\113
视频路径	光盘:\视频\113
实例必备	113.使用 Android 网络接口.pdf

9.4.1　实例说明

文件上传对于广大读者来说并不陌生，在手机中同样可以实现网站上传功能。在本节的内容中，将通过一个具体实例的实现过程，介绍在 Android 中实现文件上传的基本流程。

9.4.2　具体实现

编写主程序文件，其具体实现流程如下所示。

（1）分别声明变量 newName、uploadFile 和 actionUrl，具体代码如下所示。

```
/*变量声明
  * newName：上传后在服务器上的文件名称
  * uploadFile：要上传的文件路径
  * actionUrl：服务器上对应的程序路径*/
private String newName="image.jpg";
private String uploadFile="/data/data/irdc.example9/image.jpg";
private String actionUrl="http://127.127.0.1/upload/upload.jsp";
private TextView mText1;
private TextView mText2;
private Button mButton;
```

（2）通过 mText1 对象获取文件路径，根据 mText2 设置上传网址，单击按钮后调用上传方法 uploadFile()，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  mText1 = (TextView) findViewById(R.id.myText2);
  mText1.setText("文件路径：\n"+uploadFile);
  mText2 = (TextView) findViewById(R.id.myText3);
  mText2.setText("上传网址：\n"+actionUrl);
  /*设置 mButton 的 onClick 事件处理*/
  mButton = (Button) findViewById(R.id.myButton);
  mButton.setOnClickListener(new View.OnClickListener()
  {
    public void onClick(View v)
    {
      uploadFile();
    }
  });
}
```

（3）定义方法 uploadFile()将文件上传至 Server，具体代码如下所示。

```
/*上传文件至 Server 的方法*/
private void uploadFile()
{
  String end = "\r\n";
  String twoHyphens = "--";
  String boundary = "*****";
  try
  {
    URL url =new URL(actionUrl);
    HttpURLConnection con=(HttpURLConnection)url.openConnection();
    /*允许 Input、Output，不使用 Cache*/
    con.setDoInput(true);
    con.setDoOutput(true);
    con.setUseCaches(false);
    /*设置传送的 method=POST*/
    con.setRequestMethod("POST");
    /*setRequestProperty*/
    con.setRequestProperty("Connection", "Keep-Alive");
    con.setRequestProperty("Charset", "UTF-8");
    con.setRequestProperty("Content-Type",
                       "multipart/form-data;boundary="+boundary);
    /*设置 DataOutputStream*/
    DataOutputStream ds =
      new DataOutputStream(con.getOutputStream());
    ds.writeBytes(twoHyphens + boundary + end);
    ds.writeBytes("Content-Disposition: form-data; " +
                  "name=\"file1\";filename=\"" +
                  newName +"\"" + end);
    ds.writeBytes(end);
    /*取得文件的 FileInputStream*/
```

```
        FileInputStream fStream = new FileInputStream(uploadFile);
        /*设置每次写入 1024bytes*/
        int bufferSize = 1024;
        byte[ ] buffer = new byte[bufferSize];
        int length = -1;
        /*从文件读取数据至缓冲区*/
        while((length = fStream.read(buffer)) != -1)
        {
          /*将资料写入 DataOutputStream 中*/
          ds.write(buffer, 0, length);
        }
        ds.writeBytes(end);
        ds.writeBytes(twoHyphens + boundary + twoHyphens + end);
        fStream.close();
        ds.flush();
        /*取得 Response 内容*/
        InputStream is = con.getInputStream();
        int ch;
        StringBuffer b =new StringBuffer();
        while( ( ch = is.read() ) != -1 )
        {
          b.append( (char)ch );
        }
        /*将 Response 显示在 Dialog 对话框中*/
        showDialog(b.toString().trim());
        /*关闭 DataOutputStream*/
        ds.close();
      }
      catch(Exception e)
      {
        showDialog(""+e);
      }
    }
```

（4）定义方法 showDialog(String mess)来显示提示对话框，具体代码如下所示。

```
/*显示 Dialog 的方法*/
private void showDialog(String mess)
{
  new AlertDialog.Builder(example9.this).setTitle("Message")
   .setMessage(mess)
   .setNegativeButton("确定",new DialogInterface.OnClickListener()
   {
     public void onClick(DialogInterface dialog, int which)
     {
     }
   })
   .show();
}
```

执行后单击“上传”按钮可以将指定的文件上传到服务器，如图 9-5 所示。



图 9-5　执行效果

9.5　远程下载并安装一个软件

实例 114	远程下载并安装一个软件
源码路径	光盘:\daima\114
视频路径	光盘:\视频\114
实例必备	114.URL 和 URLConnection.pdf

9.5.1　实例说明

本实例运行后，能够远程下载指定网址的 Android 应用程序，下载到手机后打开 application installer 软件来安装这个程序。在具体实现上，先设置一个 EditText 来获取远程程序的 URL，然后通过自定义按钮打开下载程序（使用 java.net 的 URLConnection 对象来创建连接，通过 InputStream 将下载文件写入到存储卡的缓存），下载后通过自定义方法 openFile()打开文件，并根据文件扩展名判断是否为 APK 格式，是则启动内置的 Install 程序，开始安装。安装完成后，在离开 Install 时通过方法 delFile()将存储卡中的临时文件删除。

9.5.2　具体实现

编写主程序文件，其具体实现流程如下所示。

（1）单击按钮后设置将文件下载到 local 端，获取要安装程序的文件名称，具体代码如下所示。

```
mButton01.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /*文件会下载至 local 端*/
        mTextView01.setText("下载中...");
        strURL = mEditText01.getText().toString();
        /*取得欲安装程序的文件名称*/
        fileEx = strURL.substring(strURL.lastIndexOf(".")
        +1,strURL.length()).toLowerCase();
        fileNa = strURL.substring(strURL.lastIndexOf("/")
```

```
            +1,strURL.lastIndexOf("."));
            getFile(strURL);
          }
       }
    );
```

（2）如果框中的远程地址为空，则输出“请输入 URL”的提示，具体代码如下所示。

```
    mEditText01.setOnClickListener(new EditText.OnClickListener()
    {
       @Override
       public void onClick(View arg0)
       {
          // TODO Auto-generated method stub
          mEditText01.setText("");
          mTextView01.setText("远程安装程序(请输入 URL)");
       }
    });
  }
```

（3）定义方法 getFile(final String strPath)来获取下载的 URL 文件，如果有异常则输出提示，具体代码如下所示。

```
  /*处理下载 URL 文件自定义函数*/
  private void getFile(final String strPath)  {
    try
    {
       if (strPath.equals(currentFilePath) )
       {
          getDataSource(strPath);
       }
       currentFilePath = strPath;
       Runnable r = new Runnable()
       {
          public void run()
          {
             try
             {
                getDataSource(strPath);
             }
             catch (Exception e)
             {
                Log.e(TAG, e.getMessage(), e);
             }
          }
       };
       new Thread(r).start();
    }
    catch(Exception e)
    {
       e.printStackTrace();
    }
  }
```

（4）定义方法 getDataSource()来获取远程文件，主要代码如下所示。

```
 /*获取远程文件*/
private void getDataSource(String strPath) throws Exception
{
  if (!URLUtil.isNetworkUrl(strPath))
  {
    mTextView01.setText("错误的 URL");
  }
  else
  {
    /*获取 URL*/
    URL myURL = new URL(strPath);
    /*创建连接*/
    URLConnection conn = myURL.openConnection();
    conn.connect();
    /*InputStream  下载文件*/
    InputStream is = conn.getInputStream();
    if (is == null)
    {
      throw new RuntimeException("stream is null");
    }
    /*创建临时文件*/
    File myTempFile = File.createTempFile(fileNa, "."+fileEx);
    /*获取暂存盘路径*/
    currentTempFilePath = myTempFile.getAbsolutePath();
    /*将文件写入暂存盘*/
    FileOutputStream fos = new FileOutputStream(myTempFile);
    byte buf[ ] = new byte[128];
    do
    {
      int numread = is.read(buf);
      if (numread <= 0)
      {
        break;
      }
      fos.write(buf, 0, numread);
    }while (true);

    /*打开文件进行安装*/
    openFile(myTempFile);
    try
    {
      is.close();
    }
    catch (Exception ex)
    {
      Log.e(TAG, "error: " + ex.getMessage(), ex);
    }
  }
}
```

（5）定义方法 openFile(File f)设置在手机上打开文件，主要代码如下所示。

```
/*在手机上打开文件的 method*/
private void openFile(File f)
{
   Intent intent = new Intent();
   intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
   intent.setAction(android.content.Intent.ACTION_VIEW);

   /*调用 getMIMEType()获取 MimeType*/
   String type = getMIMEType(f);
   /*设置 intent 的 file 与 MimeType*/
   intent.setDataAndType(Uri.fromFile(f),type);
   startActivity(intent);
}
/*判断文件 MimeType 的 method*/
private String getMIMEType(File f)
{
   String type="";
   String fName=f.getName();
   /*获取扩展名*/
   String end=fName.substring(fName.lastIndexOf(".")
   +1,fName.length()).toLowerCase();

   /*依扩展名的类型决定 MimeType*/
   if(end.equals("m4a")||end.equals("mp3")||end.equals("mid")||
   end.equals("xmf")||end.equals("ogg")||end.equals("wav"))
   {
      type = "audio";
   }
   else if(end.equals("3gp")||end.equals("mp4"))
   {
      type = "video";
   }
   else if(end.equals("jpg")||end.equals("gif")||end.equals("png")||
   end.equals("jpeg")||end.equals("bmp"))
   {
      type = "image";
   }
   else if(end.equals("apk"))
   {
      /*android.permission.INSTALL_PACKAGES*/
      type = "application/vnd.android.package-archive";
   }
   else
   {
      type="*";
   }
   /*如果无法直接打开，则弹出软件列表供用户选择*/
   if(end.equals("apk"))
   {
   }
   else
   {
```

```
        type += "/*";
    }
    return type;
}
```

（6）定义方法 delFile()删除 SD 卡上的临时文件，主要代码如下所示。

```
/*自定义删除文件方法*/
private void delFile(String strFileName)
{
    File myFile = new File(strFileName);
    if(myFile.exists())
    {
        myFile.delete();
    }
}
```

（7）定义方法 onPause()和 onResume()，分别设置 onPause（暂停）和 onResume（重新开始）的状态，具体代码如下所示。

```
/*当 Activity 处于 onPause 状态时，更改 TextView 文字状态*/
@Override
protected void onPause()
{
    mTextView01 = (TextView)findViewById(R.id.myTextView1);
    mTextView01.setText("下载成功");
    super.onPause();
}
/*当 Activity 处于 onResume 状态时，删除临时文件*/
@Override
protected void onResume()
{
    // TODO Auto-generated method stub
    /*删除临时文件*/
    delFile(currentTempFilePath);
    super.onResume();
}
```

执行后在文本框中显示目标安装程序的路径，如图 9-6 所示。实例中的默认路径是 http://mz.ruan8.com/soft/2/sougoushoujishurufa_7786.apk，这是一个搜狗输入法程序。单击“安装”按钮后，开始下载目标文件，如图 9-7 所示。下载完成后弹出安装界面，单击 Install 按钮后开始安装，安装完成后输出提示。

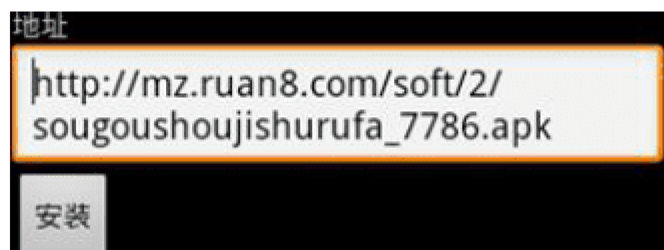


图 9-6　下载目标文件

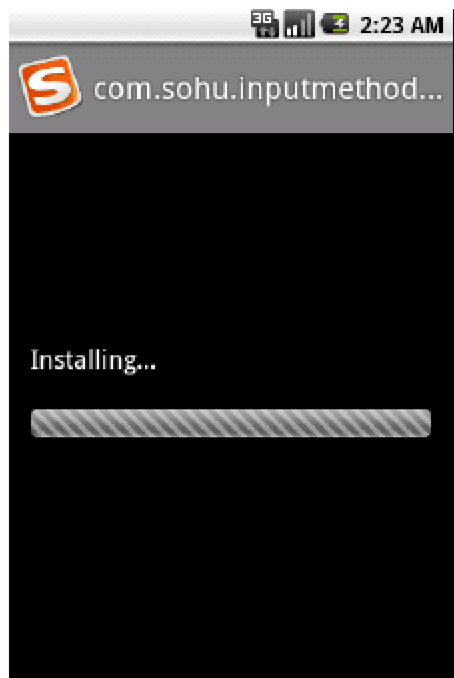


图 9-7　下载界面

9.6　移动微博发布器

实例 115	开发一个移动微博发布器
源码路径	光盘:\daima\115
视频路径	光盘:\视频\115
实例必备	115.HTTPURLConnection 详解.pdf ① 从 Internet 获取网页 ② 从 Internet 获取文件 ③ 向 Internet 发送请求参数 ④ 向 Internet 发送 XML 数据

9.6.1　实例说明

在互联网时代，使用博客的人越来越多，人们通过博客来抒发情感，记录生活中的点点滴滴，更有许多部落客，通过博客来分享不同领域的生活。为了方便人们的生活，在很多智能手机上推出了“移动博客发布器”。

RPC 是 Remote Procedure Call 的缩写，译名“远程过程调用”，XML-RPC 是一种统一标准的规范，通过 HTTP 连接的方式运行，以传送符合 XML-RPC 格式的 request 来调用远程服务器上的某个程序，进而运行博客功能。许多的网络服务业者都会以 XML-RPC 的方式提供给软件开发者一个中间媒介接口，让开发者能够根据业者定义好的方式，以 XML-RPC 的方式来使用该网站的某些功能。目前许多博客也都支持 XML-RPC 的媒介方式。

在 XML-RPC 标准中，规定 XML 内容的规则如下所示。

```
<xml version="1.0"?>
<methodCall>
  <methodName>要调用的 method name</methodName>
    <params>
    <params>参数 1</param>
    <params>参数 2</param>
    <param>参数 n</param>
    </params>
</methodCall>
```

在本实例中实现了一个“移动博客发布器”的功能，以乐多博客 http://ublog.roodo.com/为例，演示了实现从手机发布文章到乐多博客上的方法。

调用乐多博客的 metaWeblog.newPost 运行添加博客文章的动作，发出的 XML 请求内容如下所示。

```
< ?xml version="l.0"?>
<methodCall>
        <methodName >metaWeblog.newPost</methodName>
        <params>
                <param><value><string>ID</string></value></param>
                <param><value><string>账号</string></value></param>
```

```
                <param><value><string>密码</string></value></param>
                <param>
                  <value>
                     <struct>
                       <member>
                          <name>title</name>
                          <value><string>文章标题</string></value>
                       </member>
                       <member>
                          <name>descriptiori</name>
                          <value><string>内容</string></value>
                       </member>
                     </struct>
                  </value>
                </param>
                <param><value><boolean>l</boolean></value></param>
      </params>
</methodCall>
```

9.6.2 具体实现

编写主程序文件，在此文件中以 EditText 编辑框作为输入博客相关信息及文章内容的组件，当用户输入完成后单击“发布文章”按钮，此按钮的 onClick()会被触发，首先检查输入字段是否为空白，检查无误后，程序先运行 getPostString()，将输入参数转换成符合 XML-RPC 规范的 XML 格式，再调用 sendPost()将 XML 的 request 传送给相对应的博客网址，最后再取得服务器返回的 Response，并使用 Dialog 形式显示运行结果。

主程序文件的主要代码如下所示。

```
/*变量声明*/
Button mButton;
EditText mEdit1;
EditText mEdit2;
EditText mEdit3;
EditText mEdit4;
EditText mEdit5;
/*乐多博客 XML-RPC 网址*/
private String path=
   " http://blog.csdn.net/asdfg343442";
/*XML-RPC 发布文章的 method name*/
private String method="metaWeblog.newPost";

@Override
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   /*初始化对象*/
   mEdit1=(EditText)findViewById(R.id.blogId);
   mEdit2=(EditText)findViewById(R.id.blogAccount);
```

```
    mEdit3=(EditText)findViewById(R.id.blogPwd);
    mEdit4=(EditText)findViewById(R.id.artTitle);
    mEdit5=(EditText)findViewById(R.id.artContent);
    mButton=(Button)findViewById(R.id.myButton);
    /*设置发布文章的 onClick 事件*/
    mButton.setOnClickListener(new View.OnClickListener()
    {
      public void onClick(View v)
      {
        /*取得输入的信息*/
        String blogId=mEdit1.getText().toString();
        String account=mEdit2.getText().toString();
        String pwd=mEdit3.getText().toString();
        String title=mEdit4.getText().toString();
        String content=mEdit5.getText().toString();

        if(blogId.equals("")||account.equals("")||pwd.equals("")||
            title.equals("")||content.equals(""))
        {
          showDialog("没有填写内容!");
        }
        else
        {
          /*发送 XML POST 并显示 Response 内容*/
          String outS=getPostString(method,blogId,account,
                                    pwd,title,content);
          String re=sendPost(outS);
          showDialog(re);
        }
      }
    });
  }

  /*发送 request 至博客的对应网址的 method*/
  private String sendPost(String outString)
  {
    HttpURLConnection conn=null;
    String result="";
    URL url = null;
    try
    {
      url = new URL(path);
      conn = (HttpURLConnection)url.openConnection();
      /*允许 Input、Output*/
      conn.setDoInput(true);
      conn.setDoOutput(true);
      /*设置传送的 method=POST*/
      conn.setRequestMethod("POST");
      /*setRequestProperty*/
      conn.setRequestProperty("Content-Type", "text/xml");
      conn.setRequestProperty("Charset", "UTF-8");
```

```
        /*送出 request*/
        OutputStreamWriter out =
          new OutputStreamWriter(conn.getOutputStream(), "utf-8");
        out.write(outString);
        out.flush();
        out.close();
        /*解析返回的 XML 内容*/
        result=parseXML(conn.getInputStream());
        conn.disconnect();
      }
      catch(Exception e)
      {
        conn.disconnect();
        e.printStackTrace();
        showDialog(""+e);
      }
      return result;
    }

    /*解析 Response 的 XML 内容的 method*/
    private String parseXML(InputStream is)
    {
      String result="";
      Document doc = null;
      try
      {
        /*将 XML 转换成 Document 对象*/
        DocumentBuilderFactory dbf=
          DocumentBuilderFactory.newInstance();
        DocumentBuilder db=dbf.newDocumentBuilder();
        doc = db.parse(is);
        doc.getDocumentElement().normalize();
        /*检查返回值是否有包含 fault 这个 tag，有则代表发布错误*/
        int fault=doc.getElementsByTagName("fault").getLength();
        if(fault>0)
        {
          result+="发布错误!\n";
          /*取得 faultCode（错误代码）*/
          NodeList nList1=doc.getElementsByTagName("int");
          for (int i = 0; i < nList1.getLength(); ++i)
          {
            String errCode=nList1.item(i).getChildNodes().item(0)
                                  .getNodeValue();
            result+="错误代码："+errCode+"\n";
          }
          /*取得 faultString（错误信息）*/
          NodeList nList2=doc.getElementsByTagName("string");
          for (int i = 0; i < nList2.getLength(); ++i)
          {
            String errString=nList2.item(i).getChildNodes().item(0)
```

```
                                        .getNodeValue();
            result+="错误信息："+errString+"\n";
          }
        }
        else
        {
          /*发布成功，取得文章编号*/
          NodeList nList=doc.getElementsByTagName("string");
          for (int i = 0; i < nList.getLength(); ++i)
          {
            String artId=nList.item(i).getChildNodes().item(0)
                                  .getNodeValue();
            result+="发布成功!!文章编号「"+artId+"」";
          }
        }
      }
      catch (Exception ioe)
      {
        showDialog(""+ioe);
      }
      return result;
    }

    /*一组要发送的 XML 内容的 method*/
    private String getPostString(String method,String blogId,
        String account,String pwd,String title,String content)
    {
      String s="";
      s+="<methodCall>";
      s+="<methodName>"+method+"</methodName>";
      s+="<params>";
      s+="<param><value><string>"+blogId+"</string></value></param>";
      s+="<param><value><string>"+account+"</string></value></param>";
      s+="<param><value><string>"+pwd+"</string></value></param>";
      s+="<param><value><struct>";
      s+="<member><name>title</name>" +
         "<value><string>"+title+"</string></value></member>";
      s+="<member><name>description</name>" +
         "<value><string>"+content+"</string></value></member>";
      s+="</struct></value></param>";
      s+="<param><value><boolean>1</boolean></value></param>";
      s+="</params>";
      s+="</methodCall>";

      return s;
    }

    /*跳出 Dialog 的 method*/
    private void showDialog(String mess)
    {
      new AlertDialog.Builder(example174.this).setTitle("Message")
```

```
            .setMessage(mess)
            .setNegativeButton("确定", new DialogInterface.OnClickListener()
            {
              public void onClick(DialogInterface dialog, int which)
              {
              }
            })
            .show();
        }
}
```

执行后的效果如图 9-8 所示，只要拥有乐多的账号，就可以在手机上发送博客。



图 9-8　执行效果

9.7　解析和生成 XML

实例 116	在 Android 系统中解析和生成 XML
源码路径	光盘:\daima\116
视频路径	光盘:\视频\116
实例必备	116.XML 技术基础.pdf ① XML 的概述 ② XML 的语法 ③ 获取 XML 文档

9.7.1　实例说明

Android 是最常用的智能手机平台，XML 是数据交换的标准媒介，Android 中可以使用标准的 XML 生成器、解析器、转换器 API，对 XML 进行解析和转换。

9.7.2　具体实现

实现解析功能的核心文件是 SAXForHandler.java，主要实现代码如下所示。

```
public class SAXForHandler extends DefaultHandler {
	private static final String TAG = "SAXForHandler";
	private List<Person> persons;
	private String perTag ;		//通过此变量，记录前一个标签的名称
	Person person;			//记录当前 Person

	public List<Person> getPersons() {
		return persons;
	}

	//适合在此事件中触发初始化行为
	public void startDocument() throws SAXException {
		persons = new ArrayList<Person>();
		Log.i(TAG , "***startDocument()***");
	}

	public void startElement(String uri, String localName, String qName,
			Attributes attributes) throws SAXException {
		if("person".equals(localName)){
			for ( int i = 0; i < attributes.getLength(); i++ ) {
				Log.i(TAG ,"attributeName:" + attributes.getLocalName(i)
						+ "_attribute_Value:" + attributes.getValue(i));
				person = new Person();
				person.setId(Integer.valueOf(attributes.getValue(i)));
			}
		}
		perTag = localName;
		Log.i(TAG , qName+"***startElement()***");
	}

	public void characters(char[ ] ch, int start, int length) throws SAXException {
		String data = new String(ch, start, length).trim();
		if(!"".equals(data.trim())){
			Log.i(TAG ,"content: " + data.trim());
		}
		if("name".equals(perTag)){
				person.setName(data);
		}else if("age".equals(perTag)){
				person.setAge(new Short(data));
		}
	}

	public void endElement(String uri, String localName, String qName)
			throws SAXException {
		Log.i(TAG , qName+"***endElement()***");
		if("person".equals(localName)){
```

```
                    persons.add(person);
                    person = null;
            }
            perTag = null;
    }

    public void endDocument() throws SAXException {
            Log.i(TAG , "***endDocument()***");
    }
}
```

执行后的效果如图 9-9 所示。

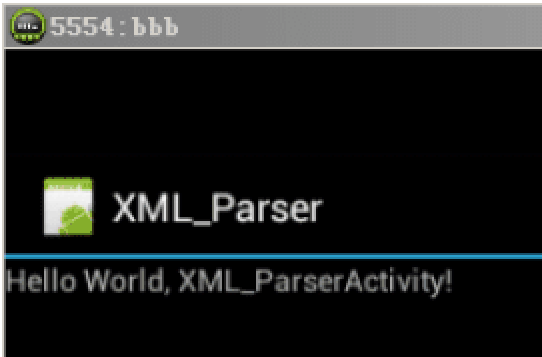


图 9-9　执行效果

9.8　获取网络中的图片

实例 117	在 Android 系统中获取网络中的图片
源码路径	光盘:\daima\117
视频路径	光盘:\视频\117
实例必备	117.使用 SAX 解析 XML 数据.pdf ① SAX 的原理 ② 基于对象和基于事件的接口

9.8.1　实例说明

在 Android 中获取网络图片是一项耗时的操作，如果直接获取有可能会出现应用程序无响应（ANR:Application Not Responding）的情况。对于这种情况，一般的方法就是耗时操作用线程来实现。

9.8.2　具体实现

先在布局文件 main.xml 中设置一个网址文本框，主要代码如下所示。

```
    <EditText
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="http://img10.360buyimg.com/book1/s75x75_g14/M0A/06/09/rBEhVVHpYGsIAAAAAAB4Ht
BqO9gAABOsAOX8xEAAHg2335.jpg"
      android:id="@+id/path"
    />
```

编写主程序文件 GetAPictureFromInternetActivity.java，主要实现代码如下所示。

```
public class GetAPictureFromInternetActivity extends Activity {
     private EditText pathText;
    private ImageView imageView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        pathText = (EditText) this.findViewById(R.id.path);
        imageView = (ImageView) this.findViewById(R.id.imageView);
    }

    public void showimage(View v){
     String path = pathText.getText().toString();
          try {
               Bitmap bitmap = ImageService.getImage(path);
               imageView.setImageBitmap(bitmap);
          } catch (Exception e) {
               e.printStackTrace();
               Toast.makeText(getApplicationContext(), R.string.error, 1).show();
          }
    }
}
```

执行后的效果如图 9-10 所示。

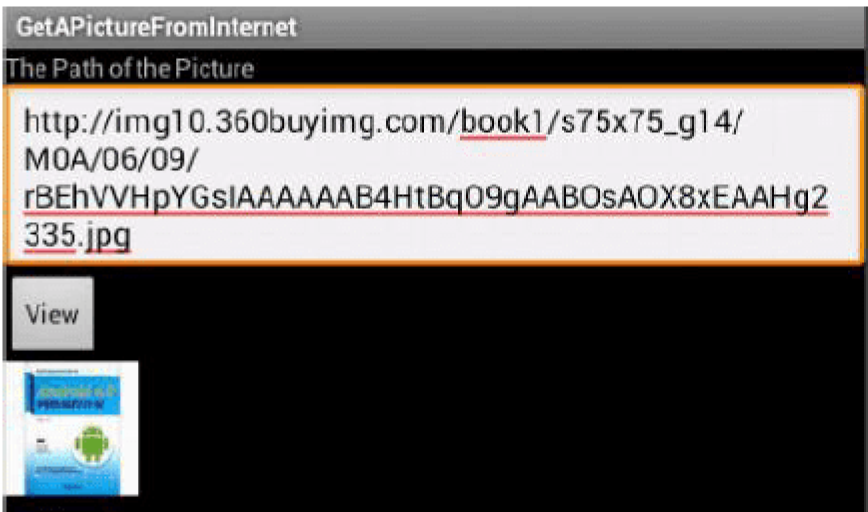


图 9-10　执行效果

9.9　获取网页的代码

实例 118	获取网络中某个网页的代码
源码路径	光盘:\daima\118
视频路径	光盘:\视频\118
实例必备	118.使用 DOM 解析 XML.pdf ① DOM 概述 ② DOM 的结构

9.9.1 实例说明

Android 在加载非本地 HTML 时，会对此 HTML 做缓存，同时会建一个数据库，用以保存 URL 地址对应的缓存文件名称，打开时间等信息，可以将 WebView 加载的 URL 地址作为查询条件获取对应的缓存文件名称，匹配出的缓存文件就是完整的 HTML 代码。

9.9.2 具体实现

编写文件 HtmlService.java，定义一个获取网页代码的业务类 HtmlService，主要代码如下所示。

```
public class HtmlService {
	/**
	 * 获取网页源码
	 * @param path 网页路径
	 * @return
	 */
	public static String getHtml(String path) throws Exception {
		HttpURLConnection conn = (HttpURLConnection)new URL(path).openConnection();
		conn.setConnectTimeout(5000);
		conn.setRequestMethod("GET");
		if(conn.getResponseCode() == 200){
			InputStream inStream = conn.getInputStream();
			byte[ ] data = StreamTool.read(inStream);
			return new String(data);
		}
		return null;
	}
}
```

编写文件 StreamTool.java，功能是将流转换为字节数组，主要代码如下所示。

```
	public static byte[ ] read(InputStream inStream) throws Exception{
		ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
		byte[ ] buffer = new byte[1024];
		int len = 0;
		while( (len = inStream.read(buffer)) != -1){
			outputStream.write(buffer, 0, len);
		}
		inStream.close();
		return outputStream.toByteArray();
	}
```

执行后的效果如图 9-11 所示。

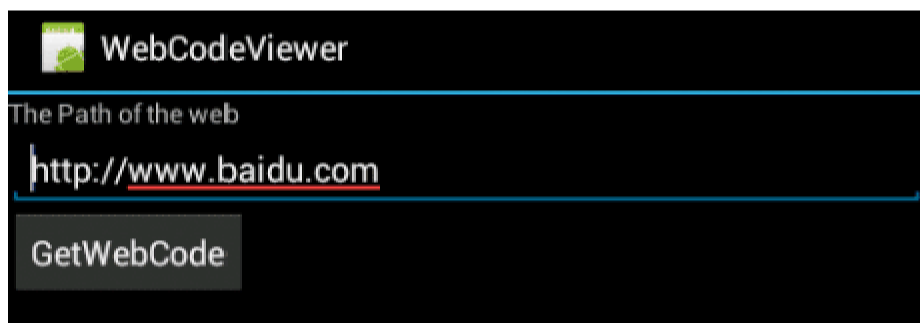


图 9-11 执行效果

第 10 章　视频和音频实战应用

在移动手机应用中，多媒体是一个重要的应用领域。从严格意义上讲，多媒体包含了屏保、图片、音频、视频和相机等应用。在本书第 8 章的内容中，已经详细介绍了图形图像应用的基本知识。本章将通过几个典型实例的实现过程，详细介绍在 Android 系统中实现视频、音频、震动、铃声实战应用的基本知识，着重讲解音频、视频、相机等应用的方法。

10.1　调节手机音量的大小

实例 119	调节手机音量的大小
源码路径	光盘:\daima\119
视频路径	光盘:\视频\119
实例必备	119.音频应用接口类介绍.pdf

10.1.1　实例说明

在使用手机时，经常需要调节音量的大小。在 Android API 中的 AudioManager 类中，提供了调节手机音量的相关方法。可以直接在程序中控制手机音量的大小，也可以切换声音的模式为震动或静音。在进行具体编码之前，需要预先准备素材图片，并将这些素材图片保存在 res\drawable 目录下。

10.1.2　具体实现

编写主程序文件，下面讲解具体实现代码。

（1）先声明系统需要的各个变量对象，具体代码如下所示。

```
/*变量声明*/
private ImageView myImage;
private ImageButton downButton;
private ImageButton upButton;
private ImageButton normalButton;
private ImageButton muteButton;
private ImageButton vibrateButton;
private ProgressBar myProgress;
private AudioManager audioMa;
private int volume=0;
```

（2）依次初始化 audioMa、myImage、myProgress、downButton、upButton、normalButton、muteButton 和 vibrateButton 变量对象，具体代码如下所示。

```
/*对象初始化*/
audioMa = (AudioManager)getSystemService(Context.AUDIO_SERVICE);
myImage = (ImageView)findViewById(R.id.myImage);
myProgress = (ProgressBar)findViewById(R.id.myProgress);
downButton = (ImageButton)findViewById(R.id.downButton);
upButton = (ImageButton)findViewById(R.id.upButton);
normalButton = (ImageButton)findViewById(R.id.normalButton);
muteButton = (ImageButton)findViewById(R.id.muteButton);
vibrateButton = (ImageButton)findViewById(R.id.vibrateButton);
```

（3）分别设置初始的手机音量大小和初始的声音模式，具体代码如下所示。

```
/*设置初始的手机音量*/
volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
myProgress.setProgress(volume);
/*设置初始的声音模式*/
int mode=audioMa.getRingerMode();
if(mode==AudioManager.RINGER_MODE_NORMAL)
{
  myImage.setImageDrawable(getResources()
                          .getDrawable(R.drawable.normal));
}
else if(mode==AudioManager.RINGER_MODE_SILENT)
{
  myImage.setImageDrawable(getResources()
                          .getDrawable(R.drawable.mute));
}
else if(mode==AudioManager.RINGER_MODE_VIBRATE)
{
  myImage.setImageDrawable(getResources()
                          .getDrawable(R.drawable.vibrate));
}
```

（4）设置单击音量调小按钮 downButton 的处理事件，每单击一次音量，调小一格，并设置调整后的声音模式，具体代码如下所示。

```
/*调小音量的 Button*/
downButton.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View arg0)
  {
    /*设置音量调小一格*/
    audioMa.adjustVolume(AudioManager.ADJUST_LOWER, 0);
    volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
    myProgress.setProgress(volume);
    /*设置调整后声音模式*/
    int mode=audioMa.getRingerMode();
    if(mode==AudioManager.RINGER_MODE_NORMAL)
    {
      myImage.setImageDrawable(getResources()
                              .getDrawable(R.drawable.normal));
    }
    else if(mode==AudioManager.RINGER_MODE_SILENT)
    {
      myImage.setImageDrawable(getResources()
```

```
                                                .getDrawable(R.drawable.mute));
        }
        else if(mode==AudioManager.RINGER_MODE_VIBRATE)
        {
            myImage.setImageDrawable(getResources()
                                                .getDrawable(R.drawable.vibrate));
        }
    }
});
```

（5）设置单击音量调大按钮 upButton 的处理事件，每单击一次音量，调大一格，并设置调整后的声音模式，具体代码如下所示。

```
/*调大音量的 Button*/
upButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        /*设置音量调大一格*/
        audioMa.adjustVolume(AudioManager.ADJUST_RAISE, 0);
        volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
        myProgress.setProgress(volume);
        /*设置调整后的声音模式*/
        int mode=audioMa.getRingerMode();
        if(mode==AudioManager.RINGER_MODE_NORMAL)
        {
            myImage.setImageDrawable(getResources()
                                                .getDrawable(R.drawable.normal));
        }
        else if(mode==AudioManager.RINGER_MODE_SILENT)
        {
            myImage.setImageDrawable(getResources()
                                                .getDrawable(R.drawable.mute));
        }
        else if(mode==AudioManager.RINGER_MODE_VIBRATE)
        {
            myImage.setImageDrawable(getResources()
                                                .getDrawable(R.drawable.vibrate));
        }
    }
});
```

（6）设置单击调整正常铃声模式按钮 normalButton 的处理事件，单击后设置铃声模式为 NORMAL，并设置音量与声音模式，具体代码如下所示。

```
/*调整铃声模式为正常模式的 Button*/
normalButton.setOnClickListener(new Button.OnClickListener()
{
    @Override
    public void onClick(View arg0)
    {
        /*设置铃声模式为 NORMAL*/
        audioMa.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
        /*设置音量与声音模式*/
        volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
```

```
            myProgress.setProgress(volume);
            myImage.setImageDrawable(getResources()
                                        .getDrawable(R.drawable.normal));
        }
    });
```

（7）设置单击调整静音铃声模式按钮 muteButton 的处理事件，首先设置铃声模式为 SILENT，然后设置音量与声音状态，具体代码如下所示。

```
    /*调整铃声模式为静音模式的 Button*/
    muteButton.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View arg0)
        {
            /*设置铃声模式为 SILENT*/
            audioMa.setRingerMode(AudioManager.RINGER_MODE_SILENT);
            /*设置音量与声音状态*/
            volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
            myProgress.setProgress(volume);
            myImage.setImageDrawable(getResources()
                                        .getDrawable(R.drawable.mute));
        }
    });
```

（8）设置单击调整震动铃声模式按钮 vibrateButton 的处理事件，首先设置铃声模式为 VIBRATE，然后设置音量与声音状态，具体代码如下所示。

```
    /*调整铃声模式为震动模式的 Button*/
    vibrateButton.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View arg0)
        {
            /*设置铃声模式为 VIBRATE*/
            audioMa.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
            /*设置音量与声音状态*/
            volume=audioMa.getStreamVolume(AudioManager.STREAM_RING);
            myProgress.setProgress(volume);
            myImage.setImageDrawable(getResources()
                                        .getDrawable(R.drawable.vibrate));
        }
    });
```

执行后将会显示一个音量调节界面，既可以设置声音模式，也可以调整音量大小，如图 10-1 所示。



图 10-1　执行效果

10.2　实现手机震动效果

实例 120	实现手机震动效果
源码路径	光盘:\daima\120
视频路径	光盘:\视频\120
实例必备	120.AudioManager 类.pdf ① 方法 ② 声音模式 ③ 基本应用 ④ 调节声音的基本步骤

10.2.1　实例说明

一款手机除了具有基本的通话功能和收发短信功能外，震动也是极为重要的功能之一。通过手机震动，能够帮助用户及时感知打来的电话或发来的短信。手机震动方式是不同的，分为一直持续震动和只震动一轮两种。在本实例中，向读者详细讲解触发手机震动事件的方法。

Android 中的震动事件 Vibration，需要设置震动的时间长短和周期，并且设置单位是毫秒。如果要建立手机震动，则必须建立 Vibratior 对象，并通过调用 vibrate 来实现震动目的。在 Vibratior 构造器中有 4 个参数，前 3 个用于设置震动大小，最后一个用于设置震动持续时间。

10.2.2　具体实现

编写主程序文件，下面详细讲解此文件的具体实现流程。

（1）设置 ToggleButton 对象来检测 ToggleButton 是否被启动，如果单击 ON 按钮则启动震动模式，如果单击 OFF 按钮则关闭震动模式，具体代码如下所示。

```
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
/*设置 ToggleButton 的对象*/
mVibrator01 = ( Vibrator )getApplication().getSystemService
(Service.VIBRATOR_SERVICE);
final ToggleButton mtogglebutton1 =
(ToggleButton) findViewById(R.id.myTogglebutton1);
final ToggleButton mtogglebutton2 =
(ToggleButton) findViewById(R.id.myTogglebutton2);
final ToggleButton mtogglebutton3 =
(ToggleButton) findViewById(R.id.myTogglebutton3);
```

（2）通过“mVibrator01.vibrate(new long[ ]{100,10,100,1000},-1);”设置短震动模式的震动周期，具体代码如下所示。

```
/*短震动*/
mtogglebutton1.setOnClickListener(new OnClickListener()
```

```
{
    public void onClick(View v)
    {
        if (mtogglebutton1.isChecked())
        {
            /*设置震动的周期*/
            mVibrator01.vibrate( new long[ ]{100,10,100,1000},-1);
            /*用 Toast 显示震动启动*/
            Toast.makeText
            (
                example.this,
                getString(R.string.str_ok),
                Toast.LENGTH_SHORT
            ).show();
        }
        else
        {
            /*取消震动*/
            mVibrator01.cancel();
            /*用 Toast 显示震动已被取消*/
            Toast.makeText
            (
                example.this,
                getString(R.string.str_end),
                Toast.LENGTH_SHORT
            ).show();
        }
    }
});
```

（3）通过“mVibrator01.vibrate(new long[ ]{100,100,100,1000},0);”设置长震动模式的震动周期，具体代码如下所示。

```
/*长震动*/
mtogglebutton2.setOnClickListener(new OnClickListener()
{
    public void onClick(View v)
    {
        if (mtogglebutton2.isChecked())
        {
            /*设置震动的周期*/
            mVibrator01.vibrate(new long[ ]{100,100,100,1000},0);

            /*用 Toast 显示震动启动*/
            Toast.makeText
            (
                example.this,
                getString(R.string.str_ok),
                Toast.LENGTH_SHORT
            ).show();
        }
        else
```

```
            {
                /*取消震动*/
                mVibrator01.cancel();

                /*用 Toast 显示震动取消*/
                Toast.makeText
                (
                    example.this,
                    getString(R.string.str_end),
                    Toast.LENGTH_SHORT
                ).show();
            }
        }
    });
```

（4）通过“mVibrator01.vibrate(new long[ ]{1000,50,1000,50,1000},0);”设置节奏震动模式的震动周期，具体代码如下所示。

```
    /*节奏震动*/
    mtogglebutton3.setOnClickListener(new OnClickListener()
    {
        public void onClick(View v)
        {
            if (mtogglebutton3.isChecked())
            {
                /*设置震动的周期*/
                mVibrator01.vibrate( new long[ ]{1000,50,1000,50,1000},0);

                /*用 Toast 显示震动启动*/
                Toast.makeText
                (
                    example.this, getString(R.string.str_ok),
                    Toast.LENGTH_SHORT
                ).show();
            }
            else
            {
                /*取消震动*/
                mVibrator01.cancel();
                /*用 Toast 显示震动取消*/
                Toast.makeText
                (
                    example.this,
                    getString(R.string.str_end),
                    Toast.LENGTH_SHORT
                ).show();
            }
        }
    });
  }
}
```

在文件 AndroidManifest.xml 中声明 Android.permission.VIBRATE 的权限，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:versionCode="1"
  android:versionName="1.0.0" package="irdc.example096">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:label="@string/app_name"
      android:name="irdc.example096.example096">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
  <uses-permission android:name="android.permission.VIBRATE" />
</manifest>
```

执行后的效果如图 10-2 所示，当选择一种模式并单击后面的图标按钮后会启动对应的震动模式，如图 10-3 所示为启动了短时间震动模式。

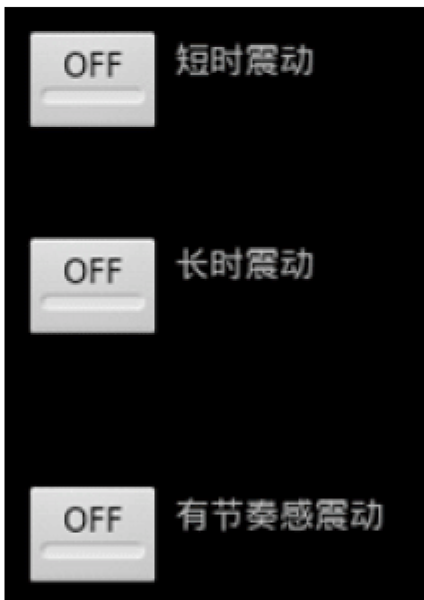


图 10-2　执行效果



图 10-3　短时间震动

10.3　手机背面朝上时自动启动震动模式

实例 121	手机背面朝上时自动启动震动模式
源码路径	光盘:\daima\121
视频路径	光盘:\视频\121
实例必备	121.录音处理.pdf ① 使用 MediaRecorder 接口录制音频 ② 使用 AudioRecord 接口录制音频

10.3.1　实例说明

通过 Android 系统中的 API，可以判断手机倾斜、旋转等模式。通过 BroadcastReceiver 对象聆听系统广播短信或 PhoneState Listener 对象，聆听系统广播的电话事件等。Android 系统的 Sensor.Manager 事件是使用 Sensor 对象实现的。为了让 Activity 程序在 onCreate()后的第一个进入点（onResume()方法）就能监视手机状态，所以在 onResume()方法中创建 IntentFilter，使用 SensorListener.registerListener()注册一个自定义的 SensorListener，使 onPause()离开程序时取消系统注册（unregisterLister）SensorListener。

因为只判断手机的倾斜或旋转状态并不够实用，所以在本实例中联合使用了 SensorListener 和 AudioManager。当程序发现手机背面朝上时，就会将铃声模式更改为震动模式。

10.3.2　具体实现

（1）编写主程序文件，在此文件中注册了 SensorListener 的 registerListener()方法，使 Activity 程序能够捕捉到 Sensor 的变化。在捕捉变化时需要传入如下 3 个参数。

- ☑ mSensorListener：SensorListener 对象，为 Activity 类成员，通过覆盖 onSensorChanged()方法作为判断。
- ☑ SensorManager.SENSOR_ORIENTATION：欲捕捉的 Sensor 事件常数。
- ☑ SensorManager.SENSOR_DELAY NORMAL：状态更改的精准度常数。

主程序文件的主要代码如下所示。

```
/*创建 SensorManager 对象*/
private SensorManager mSensorManager01;
private TextView mTextView01;

/*以私有类成员存储 AudioManager 模式*/
private int strRingerMode;

/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mTextView01 = (TextView)findViewById(R.id.myTextView1);

  /*创建 SensorManager 对象，取得 SENSOR_SERVICE 服务*/
  mSensorManager01 =
  (SensorManager)getSystemService(Context.SENSOR_SERVICE);

  /*取得现在的 AudioManager 模式*/
  GetAudioManagerMode();

  /*依据现在的 AudioManager 模式，显示于 TextView 中*/
  switch(strRingerMode)
```

```
    {
      /*正常模式*/
      case AudioManager.RINGER_MODE_NORMAL:
        mTextView01.setText(R.string.str_normal_mode);
        break;
      /*静音模式*/
      case AudioManager.RINGER_MODE_SILENT:
        mTextView01.setText(R.string.str_silent_mode);
        break;
      /*震动模式*/
      case AudioManager.RINGER_MODE_VIBRATE:
        mTextView01.setText(R.string.str_vibrate_mode);
        break;
    }
  }

  /*创建 SensorListener 捕捉 onSensorChanged 事件*/
  private final SensorListener mSensorListener =
                    new SensorListener()
  {
    @Override
    public void onSensorChanged(int sensor, float[ ] values)
    {
      // TODO Auto-generated method stub

      //float fRollAngle = values[SensorManager.DATA_X];

      /*取得 y 平面左右倾斜的 Pitch 角度*/
      float fPitchAngle = values[SensorManager.DATA_Y];

      /*正面向下（y 轴旋转），经试验，结果小于-120 为翻背面*/
      if(fPitchAngle<-120)
      {
        /*先设置为静音模式*/
        ChangeToSilentMode();

        /*再设置为震动模式*/
        ChangeToVibrateMode();

        /*判断铃声模式*/
        switch(strRingerMode)
        {
          /*正常模式*/
          case AudioManager.RINGER_MODE_NORMAL:
            mTextView01.setText(R.string.str_normal_mode);
            break;
          /*静音模式*/
          case AudioManager.RINGER_MODE_SILENT:
            mTextView01.setText(R.string.str_silent_mode);
            break;
          /*震动模式*/
```

```
          case AudioManager.RINGER_MODE_VIBRATE:
            mTextView01.setText(R.string.str_vibrate_mode);
            break;
        }
      }
      else
      {
        /*正面向上（y 轴旋转），经试验，结果大于-120 为翻正面*/
        /*更改为正常模式*/
        ChangeToNormalMode();

        /*调用更改模式后，再一次确认手机为何模式*/
        switch(strRingerMode)
        {
          case AudioManager.RINGER_MODE_NORMAL:
            mTextView01.setText(R.string.str_normal_mode);
            break;
          case AudioManager.RINGER_MODE_SILENT:
            mTextView01.setText(R.string.str_silent_mode);
            break;
          case AudioManager.RINGER_MODE_VIBRATE:
            mTextView01.setText(R.string.str_vibrate_mode);
            break;
        }
      }
    }

    @Override
    public void onAccuracyChanged(int sensor, int values)
    {
      // TODO Auto-generated method stub

    }
  };

  /*获取当下的 AudioManager 模式*/
  private void GetAudioManagerMode()
  {
    try
    {
      /*创建 AudioManager 对象，取得 AUDIO_SERVICE*/
      AudioManager audioManager =
      (AudioManager)getSystemService(Context.AUDIO_SERVICE);

      if (audioManager != null)
      {
        /*RINGER_MODE_NORMAL |
           RINGER_MODE_SILENT |
           RINGER_MODE_VIBRATE*/

        strRingerMode = audioManager.getRingerMode();
```

```
            }
        }
        catch(Exception e)
        {
            mTextView01.setText(e.toString());
            e.printStackTrace();
        }
    }

    /*更改为静音模式*/
    private void ChangeToSilentMode()
    {
        try
        {
            AudioManager audioManager =
            (AudioManager)getSystemService(Context.AUDIO_SERVICE);

            if (audioManager != null)
            {
                /*RINGER_MODE_NORMAL |
                    RINGER_MODE_SILENT |
                    RINGER_MODE_VIBRATE*/

                audioManager.setRingerMode(AudioManager.RINGER_MODE_SILENT);
                strRingerMode = audioManager.getRingerMode();
            }
        }
        catch(Exception e)
        {
            mTextView01.setText(e.toString());
            e.printStackTrace();
        }
    }

    /*更改为震动模式*/
    private void ChangeToVibrateMode()
    {
        try
        {
            AudioManager audioManager =
            (AudioManager)getSystemService(Context.AUDIO_SERVICE);

            if (audioManager != null)
            {
                /*调用 setRingerMode()方法，设置模式*/
                audioManager.setRingerMode
                (
                    AudioManager.RINGER_MODE_VIBRATE
                );
                /*RINGER_MODE_NORMAL |
                    RINGER_MODE_SILENT |
```

```
            RINGER_MODE_VIBRATE*/
        strRingerMode = audioManager.getRingerMode();
      }
    }
    catch(Exception e)
    {
      mTextView01.setText(e.toString());
      e.printStackTrace();
    }
  }

  /*更改为正常模式*/
  private void ChangeToNormalMode()
  {
    try
    {
      AudioManager audioManager =
      (AudioManager)getSystemService(Context.AUDIO_SERVICE);

      if (audioManager != null)
      {
        /*RINGER_MODE_NORMAL |
            RINGER_MODE_SILENT |
            RINGER_MODE_VIBRATE*/
        audioManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
        strRingerMode = audioManager.getRingerMode();
      }
    }
    catch(Exception e)
    {
      mTextView01.setText(e.toString());
      e.printStackTrace();
    }
  }

  @Override
  protected void onResume()
  {
    // TODO Auto-generated method stub

    /*注册一个 SensorListener 的 Listener*/
    /*传入 Sensor 模式与 rate*/
    mSensorManager01.registerListener
    (
      mSensorListener,
      SensorManager.SENSOR_ORIENTATION,
      SensorManager.SENSOR_DELAY_NORMAL
    );
    super.onResume();
  }

  @Override
```

```
    protected void onPause()
    {
        // TODO Auto-generated method stub

        /*覆盖 onPause 事件，取消 mSensorListener*/
        mSensorManager01.unregisterListener(mSensorListener);
        super.onPause();
    }
}
```

（2）编写文件 AndroidManifest.xml，在此声明 Android.permission.VIBRATE 权限，主要代码如下所示。

```
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
```

在真机中执行上述代码后，如果将手机反转则会自动进入震动模式。

10.4　在手机中播放 MP3 文件

实例 122	在手机中播放 MP3 文件
源码路径	光盘:\daima\122
视频路径	光盘:\视频\122
实例必备	122.播放音频.pdf ① 使用 AudioTrack 播放音频 ② 使用 MediaPlayer 播放音频

10.4.1　实例说明

使用手机时，经常需要播放 MP3 文件。本实例中插入了 3 个按钮，分别用于播放、暂停和停止 MP3 音乐。当单击“播放”按钮后会从手机资源中获取 MP3 文件并播放。在具体实现上，先添加一个 MediaPlayer 对象，并使用方法 MediaPlayer.creat()创建播放器资源，然后分别通过方法 MediaPlay.creat()、MediaPlay.stop()和 MediaPlay.pause()实现开始播放、停止播放和暂停播放功能。为了处理按钮所需要的各个事件，需要覆盖各 ImageButton 的 onClick()，用于通过按钮来控制 MediaPlayer 的状态。

10.4.2　具体实现

编写主程序文件，具体实现流程如下所示。

（1）分别声明 ImageButton、TextView 和 MediaPlayer 对象变量，使用变量 Flag 确认音乐是否暂停，其默认值为 false，具体代码如下所示。

```
    /*声明 ImageButton、TextView、MediaPlayer 变量*/
    private ImageButton mButton01, mButton02, mButton03;
    private TextView mTextView01;
    private MediaPlayer mMediaPlayer01;
    /*声明一个 Flag 作为确认音乐是否暂停的变量并默认为 false*/
    private boolean bIsPaused = false;
```

（2）单击“播放”按钮后，先在 MediaPlayer 中获取播放资源，然后开始播放，并同时改变 TextView 的状态为开始播放，具体代码如下所示。

```
/*单击“播放”按钮*/
mButton01.setOnClickListener(new ImageButton.OnClickListener()
{
  @Override
  /*覆盖 OnClick 事件*/
  public void onClick(View v)
  {
    try
    {

      if (mMediaPlayer01 != null)
      {
        mMediaPlayer01.stop();
      }
      /*在 MediaPlayer 中取得播放资源与 stop()之后
       *在准备 Playback 的状态前一定要使用 MediaPlayer.prepare()*/
      mMediaPlayer01.prepare();
      /*开始或恢复播放*/
      mMediaPlayer01.start();
      /*改变 TextView 为开始播放状态*/
      mTextView01.setText(R.string.str_start);
    }
    catch (Exception e)
    {
      // TODO Auto-generated catch block
      mTextView01.setText(e.toString());
      e.printStackTrace();
    }
  }
});
```

（3）编写单击停止播放的处理事件，通过方法 mMediaPlayer01.stop()停止播放 MP3 文件，改变 TextView 为停止播放状态，具体代码如下所示。

```
/*停止播放*/
mButton02.setOnClickListener(new ImageButton.OnClickListener()
{
  @Override
  public void onClick(View arg0)
  {
    // TODO Auto-generated method stub
    try
    {
      if (mMediaPlayer01 != null)
      {
        /*停止播放*/
        mMediaPlayer01.stop();
        /*改变 TextView 为停止播放状态*/
        mTextView01.setText(R.string.str_close);
      }
```

```
        }
        catch (Exception e)
        {
          //TODO Auto-generated catch block
          mTextView01.setText(e.toString());
          e.printStackTrace();
        }
      }
    });
```

（4）编写单击“暂停”按钮的处理事件，首先判断是否处于暂停状态，如果不是则使其暂停，具体代码如下所示。

```
    /*暂停播放*/
    mButton03.setOnClickListener(new ImageButton.OnClickListener()
    {
      public void onClick(View arg0)
      {
        // TODO Auto-generated method stub
        try
        {
          if (mMediaPlayer01 != null)
          {
            /*是否为暂停状态？否*/
            if(bIsPaused==false)
            {
              /*暂停播放*/
              mMediaPlayer01.pause();
              /*设置 Flag 为 true 表示 Player 状态为暂停*/
              bIsPaused = true;
              /*改变 TextView 为暂停播放*/
              mTextView01.setText(R.string.str_pause);
            }
            /*是否为暂停状态？是*/
            else if(bIsPaused==true)
            {
              /*恢复播放状态*/
              mMediaPlayer01.start();
              /*设置 Flag 为 false，表示 Player 状态为非暂停状态*/
              bIsPaused = false;
              /*改变 TextView 为开始播放*/
              mTextView01.setText(R.string.str_start);
            }
          }
        }
        catch (Exception e)
        {
          // TODO Auto-generated catch block
          mTextView01.setText(e.toString());
          e.printStackTrace();
        }
      }
    });
```

（5）定义 MediaPlayer.OnCompletionLister 来监听文件是否播放完毕，完毕后改变 TextView 状态为播放结束，具体代码如下所示。

```
/*监听是否播放完毕*/
mMediaPlayer01.setOnCompletionListener(
  new MediaPlayer.OnCompletionListener()
{
  // @Override
  /*覆盖文件播放完毕事件*/
  public void onCompletion(MediaPlayer arg0)
  {
    try
    {
      /*解除资源与 MediaPlayer 的赋值关系
       * 使资源可以为其他程序利用*/
      mMediaPlayer01.release();
      /*改变 TextView 为播放结束*/
      mTextView01.setText(R.string.str_OnCompletionListener);
    }
    catch (Exception e)
    {
      mTextView01.setText(e.toString());
      e.printStackTrace();
    }
  }
});
```

（6）编写 MediaPlayer.OnErrorListener 来监听是否出现错误，当发生错误时解除资源与 MediaPlayer 的赋值，具体代码如下所示。

```
/*监听是否出现播放错误*/
mMediaPlayer01.setOnErrorListener(new MediaPlayer.OnErrorListener()
{
  @Override
  /*覆盖错误处理事件*/
  public boolean onError(MediaPlayer arg0, int arg1, int arg2)
  {
    // TODO Auto-generated method stub
    try
    {
      /*发生错误时也解除资源与 MediaPlayer 的赋值*/
      mMediaPlayer01.release();
      mTextView01.setText(R.string.str_OnErrorListener);
    }
    catch (Exception e)
    {
      mTextView01.setText(e.toString());
      e.printStackTrace();
    }
    return false;
  }
});
```

执行后在屏幕中通过 3 个按钮播放指定的 MP3 文件，如图 10-4 所示。



图 10-4　执行效果

10.5　编写一个录音程序

实例 123	编写一个录音程序
源码路径	光盘:\daima\123
视频路径	光盘:\视频\123
实例必备	123.使用 SoundPool 播放音频.pdf ① 主要特点 ② 载入音效的方法 ③ 使用流程 ④ 安装 ADT ⑤ 设定 Android SDK Home ⑥ 验证开发环境 ⑦ 创建 Android 虚拟设备（AVD） ⑧ 启动 AVD 模拟器

10.5.1　实例说明

在手机应用中，录音是一个十分重要的功能。在本实例中插入了 4 个按钮，分别实现录音、停止录音、播放录音和删除录音这 4 种操作。为了能够不限制录音的长度，现将录音暂时保存到存储卡，当录音完毕后，再将录音文件显示在 ListView。单击文件后，可以播放或删除录音文件。

10.5.2　具体实现

编写主程序文件，其具体实现流程如下所示。

（1）分别构造 4 个按钮对象和两个文本对象，然后设置按钮状态不可选，具体代码如下所示。

```
/** Called when the activity is first created.*/
@Override
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   /*设置 4 个按钮和两个文本*/
   myButton1 = (ImageButton) findViewById(R.id.ImageButton01);
   myButton2 = (ImageButton) findViewById(R.id.ImageButton02);
   myButton3 = (ImageButton) findViewById(R.id.ImageButton03);
```

```
myButton4 = (ImageButton) findViewById(R.id.ImageButton04);
myListView1 = (ListView) findViewById(R.id.ListView01);
myTextView1 = (TextView) findViewById(R.id.TextView01);
/*设置按钮状态为不可选*/
myButton2.setEnabled(false);
myButton3.setEnabled(false);
myButton4.setEnabled(false);
```

（2）通过方法 sdCardExit()判断是否插入 SD 卡，然后将获取的 SD 卡路径作为录音的文件位置，并取得 SD 卡目录中的所有.amr 格式的文件，最后将 ArrayAdapter 添加到 ListView 对象中以列表显示录音文件，具体代码如下所示。

```
/*判断是否插入 SD 卡*/
sdCardExit = Environment.getExternalStorageState().equals(
      android.os.Environment.MEDIA_MOUNTED);
/*获取 SD 卡路径作为录音的文件位置*/
if (sdCardExit)
   myRecAudioDir = Environment.getExternalStorageDirectory();
/*获取 SD 卡目录里的所有.amr 文件*/
getRecordFiles();
adapter = new ArrayAdapter<String>(this,
      R.layout.my_simple_list_item, recordFiles);
/*将 ArrayAdapter 添加到 ListView 对象中*/
myListView1.setAdapter(adapter);
```

（3）编写单击“录音”按钮后的录音处理事件，先创建录音文件，然后设置录音来源为麦克风，最后通过 myTextView1.setText("录音中")设置录音过程显示的提示文本，具体代码如下所示。

```
/*单击“录音”按钮的处理事件*/
myButton1.setOnClickListener(new ImageButton.OnClickListener()
{
   @Override
   public void onClick(View arg0)
   {
      try
      {
         if (!sdCardExit)
         {
            Toast.makeText(example177.this, "请插入 SD Card",
                  Toast.LENGTH_LONG).show();
            return;
         }
         /*创建录音文件*/
         myRecAudioFile = File.createTempFile(strTempFile, ".amr",
               myRecAudioDir);
         mMediaRecorder01 = new MediaRecorder();
         /*设置录音来源为麦克风*/
         mMediaRecorder01
               .setAudioSource(MediaRecorder.AudioSource.MIC);
         mMediaRecorder01
               .setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
         mMediaRecorder01
               .setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
```

```
                mMediaRecorder01.setOutputFile(myRecAudioFile
                    .getAbsolutePath());
                mMediaRecorder01.prepare();
                mMediaRecorder01.start();
                myTextView1.setText("录音中");
                myButton2.setEnabled(true);
                myButton3.setEnabled(false);
                myButton4.setEnabled(false);
                isStopRecord = false;
            }
            catch (IOException e)
            {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });
```

（4）编写单击“停止”按钮的处理事件，首先使用方法 mMediaRecorder01.stop()停止录音，然后将录音文件名传递给 Adapter，具体代码如下所示。

```
    /*停止*/
    myButton2.setOnClickListener(new ImageButton.OnClickListener()
    {
        @Override
        public void onClick(View arg0)
        {
            // TODO Auto-generated method stub
            if (myRecAudioFile != null)
            {
                /*停止录音*/
                mMediaRecorder01.stop();
                mMediaRecorder01.release();
                mMediaRecorder01 = null;
                /*将录音频文件名传给 Adapter*/
                adapter.add(myRecAudioFile.getName());
                myTextView1.setText("停止：" + myRecAudioFile.getName());
                myButton2.setEnabled(false);
                isStopRecord = true;
            }
        }
    });
```

（5）编写单击“播放”按钮的处理事件，单击后打开播放的程序，主要代码如下所示。

```
    /*播放*/
    myButton3.setOnClickListener(new ImageButton.OnClickListener()
    {
        @Override
        public void onClick(View arg0)
        {
            // TODO Auto-generated method stub
            if (myPlayFile != null && myPlayFile.exists())
            {
```

```
            /*打开播放的程序*/
            openFile(myPlayFile);
          }
        }
      });
```

（6）编写单击“删除”按钮的处理事件，首先在 Adapter 中删除录音文件名，然后删除这个文件的具体资源，具体代码如下所示。

```
      /*删除事件*/
      myButton4.setOnClickListener(new ImageButton.OnClickListener()
      {
        @Override
        public void onClick(View arg0)
        {
          if (myPlayFile != null)
          {
            /*先将 Adapter 删除文件名*/
            adapter.remove(myPlayFile.getName());
            /*删除文件*/
            if (myPlayFile.exists())
              myPlayFile.delete();
            myTextView1.setText("完成删除");
          }
        }
      });
```

（7）编写单击 Adapter 列表中某个录制文件的处理事件，如果有文件，单击后将删除并将播放按钮设置为 Enable 不可用，然后输出选择提示语句。具体代码如下所示。

```
      myListView1.setOnItemClickListener
      (new AdapterView.OnItemClickListener()
      {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1,
            int arg2, long arg3)
        {
          /*在单击文件名时将“删除”和“播放”按钮设为 Enable*/
          myButton3.setEnabled(true);
          myButton4.setEnabled(true);
          myPlayFile = new File(myRecAudioDir.getAbsolutePath()
              + File.separator
              + ((CheckedTextView) arg1).getText());
          myTextView1.setText("你选的是："
              + ((CheckedTextView) arg1).getText());
        }
      });
```

（8）定义方法 onStop()实现停止录音操作，具体代码如下所示。

```
    protected void onStop()
    {
      if (mMediaRecorder01 != null && !isStopRecord)
      {
        /*停止录音*/
```

```
            mMediaRecorder01.stop();
            mMediaRecorder01.release();
            mMediaRecorder01 = null;
        }
        super.onStop();
    }
```

（9）定义方法 getRecordFiles()来获取文件的长度，在此设置只能获取.amr 格式的文件，具体代码如下所示。

```
    private void getRecordFiles()
    {
        recordFiles = new ArrayList<String>();
        if (sdCardExit)
        {
            File files[ ] = myRecAudioDir.listFiles();
            if (files != null)
            {
                for (int i = 0; i < files.length; i++)
                {
                    if (files[i].getName().indexOf(".") >= 0)
                    {
                        /*只取.amr 文件*/
                        String fileS = files[i].getName().substring(
                                files[i].getName().indexOf("."));
                        if (fileS.toLowerCase().equals(".amr"))
                            recordFiles.add(files[i].getName());
                    }
                }
            }
        }
    }
```

（10）定义方法 openFile(File f)打开播放指定的录音文件，主要代码如下所示。

```
    /*打开播放录音文件的程序*/
    private void openFile(File f)
    {
        Intent intent = new Intent();
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.setAction(android.content.Intent.ACTION_VIEW);
        String type = getMIMEType(f);
        intent.setDataAndType(Uri.fromFile(f), type);
        startActivity(intent);
    }
```

（11）定义方法 getMIMEType(File f)设置系统可接受的文件类型，在此设置 audio 类型、image 类型和其他类型，具体代码如下所示。

```
    private String getMIMEType(File f)
    {
        String end = f.getName().substring(
                f.getName().lastIndexOf(".") + 1, f.getName().length())
                .toLowerCase();
        String type = "";
```

```
    if (end.equals("mp3") || end.equals("aac") || end.equals("aac")
            || end.equals("amr") || end.equals("mpeg")
            || end.equals("mp4"))
    {
        type = "audio";
    }
    else if (end.equals("jpg") || end.equals("gif")
            || end.equals("png") || end.equals("jpeg"))
    {
        type = "image";
    }
    else
    {
        type = "*";
    }
    type += "/*";
    return type;
}
```

还需在文件 AndroidManifest.xml 中声明录音权限，具体代码如下所示。

```
<uses-permission android:name="android.permission.RECORD_AUDIO">
```

至此，整个实例介绍完毕，执行后的效果如图 10-5 所示。当单击“录音”按钮时开始录音，如图 10-6 所示。当单击“停止”按钮后停止录音，并在列表中显示录制的音频文件。当选中音频文件并单击“删除”按钮后会删除选中的音频文件，单击“播放”按钮后会播放选中的音频文件。



图 10-5　执行效果



图 10-6　正在录音

10.6　实现相机预览和拍照功能

实例 124	在手机中实现相机预览和拍照功能
源码路径	光盘:\daima\124
视频路径	光盘:\视频\124
实例必备	124.使用 Ringtone 播放铃声.pdf

10.6.1　实例说明

手机中拍照和录制视频也是十分重要的功能，在手机相机中可以通过 Preview 实现预览功能。本实例实现一个简单的拍照功能，在实例中以 Activity 为基础，在 Layout 中配置了 3 个按钮，分别实现打开预览、关闭相机和拍照处理功能。当单击“拍照”按钮后会将拍到的画面截取下来并存储到 SD 卡中，然后将拍下来的图片显示在 Activity 的 ImageView 控件中。为避免拍摄相片造成的存储卡垃圾暂存堆栈，在退出程序前应删除临时文件。

10.6.2 具体实现

编写主程序文件，其具体实现流程如下所示。

（1）引用 PictureCallback 作为拍照后的事件，具体代码如下所示。

```
/*引用 Camera 类*/
import android.hardware.Camera;

/*引用 PictureCallback 作为拍照后的事件*/
import android.hardware.Camera.PictureCallback;
import android.hardware.Camera.ShutterCallback;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
```

（2）创建私有 Camera 对象，然后分别创建 mImageView01、mTextView01、TAG、mSurfaceView01、mSurfaceHolder01 和 intScreenY 作为预览相片用，具体代码如下所示。

```
/*使 Activity 实现 SurfaceHolder.Callback*/
public class example10 extends Activity
implements SurfaceHolder.Callback
{
  /*创建私有 Camera 对象*/
  private Camera mCamera01;
  private Button mButton01, mButton02, mButton03;

  /*作为 review 已拍相片之用*/
  private ImageView mImageView01;
  private TextView mTextView01;
  private String TAG = "HIPPO";
  private SurfaceView mSurfaceView01;
  private SurfaceHolder mSurfaceHolder01;
  //private int intScreenX, intScreenY;
```

（3）设置默认相机预览模式为 false，将照下来的图片存储为\sdcard\camera_snap.jpg，具体代码如下所示。

```
  /*默认相机预览模式为 false*/
  private boolean bIfPreview = false;

  /*将拍摄的图片存储在此*/
  private String strCaptureFilePath = "/sdcard/camera_snap.jpg";
```

（4）使用 requestWindowFeature 设置全屏幕运行，然后判断存储卡是否存在，如果不存在则提醒用户未安装存储卡，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);

  /*使应用程序全屏幕运行，不使用 title bar*/
  requestWindowFeature(Window.FEATURE_NO_TITLE);
  setContentView(R.layout.main);

  /*判断存储卡是否存在*/
  if(!checkSDCard())
  {
    /*提醒 User 未安装存储卡*/
    mMakeTextToast
    (
      getResources().getText(R.string.str_err_nosd).toString(),
      true
    );
  }
```

（5）通过 DisplayMetrics 对象 dm 获取屏幕解析像素，然后以 SurfaceView 作为相机预览用，绑定 SurfaceView 后获取 SurfaceHolder 对象，通过 setFixedSize 可以设置预览大小，具体代码如下所示。

```
  /*取得屏幕解析像素*/
  DisplayMetrics dm = new DisplayMetrics();
  getWindowManager().getDefaultDisplay().getMetrics(dm);
  mTextView01 = (TextView) findViewById(R.id.myTextView1);
  mImageView01 = (ImageView) findViewById(R.id.myImageView1);
  /*以 SurfaceView 作为相机 Preview 之用*/
  mSurfaceView01 = (SurfaceView) findViewById(R.id.mSurfaceView1);

  /*绑定 SurfaceView，取得 SurfaceHolder 对象*/
  mSurfaceHolder01 = mSurfaceView01.getHolder();
  /*Activity 必须实现 SurfaceHolder.Callback*/
  mSurfaceHolder01.addCallback(example10.this);

  /*额外设置预览大小，在此不使用*/
  /*以 SURFACE_TYPE_PUSH_BUFFERS(3)作为 SurfaceHolder 显示类型*/
  mSurfaceHolder01.setType
  (SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

  mButton01 = (Button)findViewById(R.id.myButton1);
  mButton02 = (Button)findViewById(R.id.myButton2);
  mButton03 = (Button)findViewById(R.id.myButton3);
```

（6）编写打开相机和预览按钮事件，自定义初始化打开相机函数，具体代码如下所示。

```
  /*打开相机及 Preview*/
  mButton01.setOnClickListener(new Button.OnClickListener()
  {
    @Override
    public void onClick(View arg0)
    {
      // TODO Auto-generated method stub
```

```
        /*自定义初始化打开相机函数*/
        initCamera();
      }
    });
```

（7）设置停止预览按钮事件，自定义重置相机并关闭相机预览函数，具体代码如下所示。

```
    /*停止 Preview 及相机*/
    mButton02.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        /*自定义重置相机，并关闭相机预览函数*/
        resetCamera();
      }
    });
```

（8）设置停止拍照按钮事件，存储卡存在时才允许拍照，自定义函数 takePicture 实现拍照功能，具体代码如下所示。

```
    /*拍照*/
    mButton03.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View arg0)
      {
        // TODO Auto-generated method stub

        /*当存储卡存在才允许拍照，存储暂存图像文件*/
        if(checkSDCard())
        {
          /*自定义拍照函数*/
          takePicture();
        }
        else
        {
          /*存储卡不存在显示提示*/
          mTextView01.setText
          (
            getResources().getText(R.string.str_err_nosd).toString()
          );
        }
      }
    });
```

（9）定义方法 initCamera()，如果相机处于非预览模式则打开相机，具体代码如下所示。

```
  /*自定义初始相机函数*/
  private void initCamera()
  {
    if(!bIfPreview)
    {
      /*若相机不在预览模式，则打开相机*/
      mCamera01 = Camera.open();
    }
```

```
    if (mCamera01 != null && !bIfPreview)
    {
      Log.i(TAG, "inside the camera");

      /*创建 Camera.Parameters 对象*/
      Camera.Parameters parameters = mCamera01.getParameters();

      /*设置相片格式为 JPEG*/
      parameters.setPictureFormat(PixelFormat.JPEG);

      /*指定 Preview 的屏幕大小*/
      parameters.setPreviewSize(320, 240);

      /*设置图片分辨率大小*/
      parameters.setPictureSize(320, 240);

      /*将 Camera.Parameters 设置于 Camera*/
      mCamera01.setParameters(parameters);

      /*setPreviewDisplay 唯一的参数为 SurfaceHolder*/
      mCamera01.setPreviewDisplay(mSurfaceHolder01);

      /*立即运行 Preview*/
      mCamera01.startPreview();
      bIfPreview = true;
    }
  }
```

（10）定义方法 takePicture()实现拍照并撷取图像，具体代码如下所示。

```
  /*拍照并撷取图像*/
  private void takePicture()
  {
    if (mCamera01 != null && bIfPreview)
    {
      /*调用 takePicture()方法拍照*/
      mCamera01.takePicture
      (shutterCallback, rawCallback, jpegCallback);
    }
  }
```

（11）定义方法 resetCamera()实现相机重置，具体代码如下所示。

```
  /*相机重置*/
  private void resetCamera()
  {
    if (mCamera01 != null && bIfPreview)
    {
      mCamera01.stopPreview();
      /*扩展学习，释放 Camera 对象*/
      //mCamera01.release();
      mCamera01 = null;
      bIfPreview = false;
    }
```

```
}

private ShutterCallback shutterCallback = new ShutterCallback()
{
   public void onShutter()
   {
      // Shutter has closed
   }
};

private PictureCallback rawCallback = new PictureCallback()
{
   public void onPictureTaken(byte[ ] _data, Camera _camera)
   {
      // TODO Handle RAW image data
   }
};
```

（12）定义方法 delFile(String strFileName)自定义删除临时文件，具体代码如下所示。

```
/*自定义删除文件函数*/
private void delFile(String strFileName)
{
   try
   {
      File myFile = new File(strFileName);
      if(myFile.exists())
      {
         myFile.delete();
      }
   }
   catch (Exception e)
   {
      Log.e(TAG, e.toString());
      e.printStackTrace();
   }
}
```

（13）定义方法 mMakeTextToast(String str, boolean isLong)输出提示语句，具体代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)
{
   if(isLong==true)
   {
      Toast.makeText(example.this, str, Toast.LENGTH_LONG).show();
   }
   else
   {
      Toast.makeText(example.this, str, Toast.LENGTH_SHORT).show();
   }
}
```

（14）定义方法 checkSDCard()检查是否有存储卡，具体代码如下所示。

```
private boolean checkSDCard()
{
```

```
    /*判断存储卡是否存在*/
    if(android.os.Environment.getExternalStorageState().equals
    (android.os.Environment.MEDIA_MOUNTED))
    {
      return true;
    }
    else
    {
      return false;
    }
  }
  public void surfaceChanged
  (SurfaceHolder surfaceholder, int format, int w, int h)
  {
    Log.i(TAG, "Surface Changed");
  }

  public void surfaceCreated(SurfaceHolder surfaceholder)
  {
    // TODO Auto-generated method stub
    Log.i(TAG, "Surface Changed");
  }
```

在文件 AndroidManifest.xml 中声明 android.permission.CAMERA 权限，具体代码如下所示。

```
<uses-permission android:name="android.permission.CAMERA">
```

执行后的效果如图 10-7 所示，分别单击“预览”“拍照”“关闭”按钮后可以实现对应的功能。

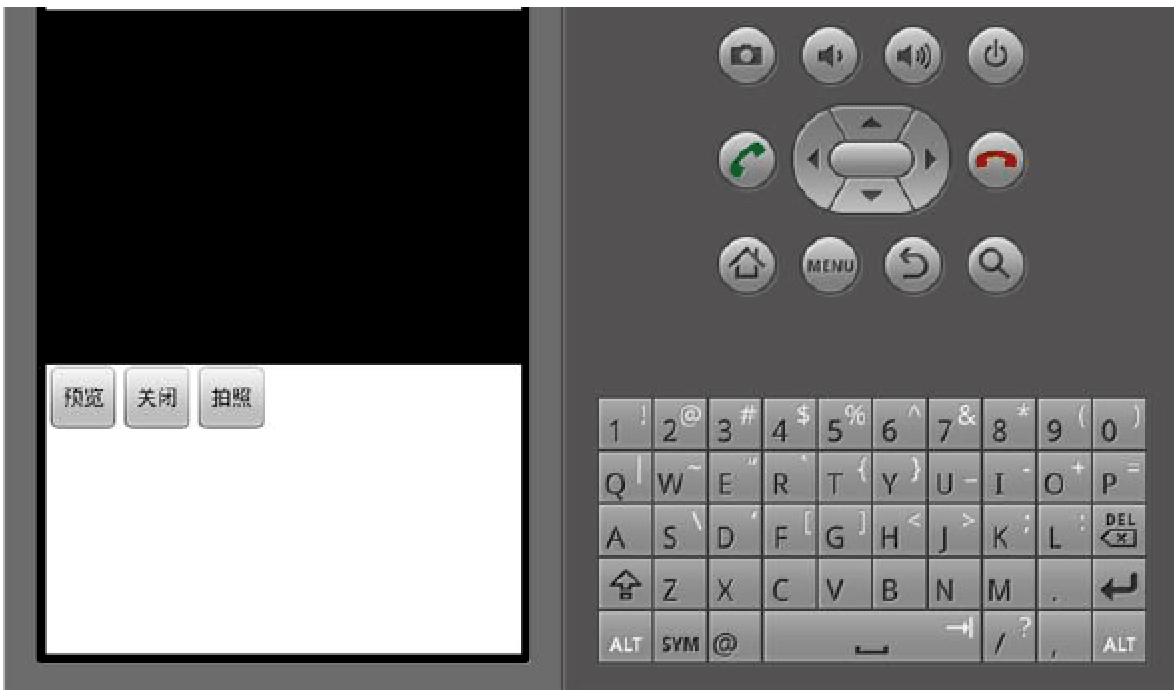


图 10-7　执行效果

10.7　在手机中播放影片

实例 125	在手机中播放影片
源码路径	光盘:\daima\125
视频路径	光盘:\视频\125
实例必备	125.使用 JetPlayer 播放音频.pdf

10.7.1　实例说明

在 Android 系统中内置了 VideoView Widget 作为多媒体视频播放器。VideoView Widget 和前面介绍的 Widget 使用方法类似，必须先在 Layout XML 中定义 VideoView 属性，在程序中通过 findViewById() 方法即可创建 VideoView 对象。在本实例中，预先准备了两个.3gp 格式的视频文件，然后将这两个文件上传到虚拟 SD 卡中。然后插入两个按钮，单击按钮分别实现对这两个视频文件的播放。

10.7.2　具体实现

编写主程序文件，具体实现流程如下所示。

（1）设置默认判别是否安装存储卡 flag 值为 false，然后设置全屏幕显示，具体代码如下所示。

```
/*默认判别是否安装存储卡 flag 为 false*/
private boolean bIfSDExist = false;
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);

  /*全屏幕*/
  getWindow().setFormat(PixelFormat.TRANSLUCENT);
  setContentView(R.layout.main);
```

（2）判断存储卡是否存在，不存在则通过 mMakeTextToast 输出提示，具体代码如下所示。

```
  /*判断存储卡是否存在*/
  if(android.os.Environment.getExternalStorageState().equals
  (android.os.Environment.MEDIA_MOUNTED))
  {
    bIfSDExist = true;
  }
  else
  {
    bIfSDExist = false;
    mMakeTextToast
    (
      getResources().getText(R.string.str_err_nosd).toString(),
      true
    );
  }
```

（3）定义单击第一个按钮的处理事件，通过函数 playVideo(strVideoPath)来播放第一个影片，具体代码如下所示。

```
  mButton01.setOnClickListener(new Button.OnClickListener()
  {
    @Override
    public void onClick(View arg0)
    {
      // TODO Auto-generated method stub
      if(bIfSDExist)
      {
```

```
                /*播放影片路径 1*/
                strVideoPath = "file:///sdcard/hello.3gp";
                playVideo(strVideoPath);
            }
        }
    });
```

（4）定义单击第二个按钮的处理事件，通过函数 playVideo(strVideoPath)来播放第二个影片，具体代码如下所示。

```
    mButton02.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View arg0)
        {
            // TODO Auto-generated method stub
            if(bIfSDExist)
            {
                /*播放影片路径 2*/
                strVideoPath = "file:///sdcard/test.3gp";
                playVideo(strVideoPath);
            }
        }
    });
```

（5）定义方法 playVideo()，使用 VideoView 来播放指定路径的影片，具体代码如下所示。

```
/*自定义以 VideoView 播放影片*/
private void playVideo(String strPath)
{
    if(strPath!="")
    {
        /*调用 VideoURI()方法，指定解析路径*/
        mVideoView01.setVideoURI(Uri.parse(strPath));

        /*设置控制 Bar 显示于此 Context 中*/
        mVideoView01.setMediaController
        (new MediaController(example179.this));

        mVideoView01.requestFocus();

        /*调用 VideoView.start()自动播放*/
        mVideoView01.start();
        if(mVideoView01.isPlaying())
        {
            /*以下程序不会被运行，因 start()后还需要 preparing()*/
            mTextView01.setText("Now Playing:"+strPath);
            Log.i(TAG, strPath);
        }
    }
}
```

（6）定义方法 mMakeTextToast()来输出提醒语句，具体代码如下所示。

```
public void mMakeTextToast(String str, boolean isLong)
{
```

```
    if(isLong==true)
    {
      Toast.makeText(example179.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
      Toast.makeText(example179.this, str, Toast.LENGTH_SHORT).show();
    }
  }
```

执行后的效果如图 10-8 所示，当单击“播放影片 1”和“播放影片 2”按钮后分别播放预设的影片。



图 10-8　执行效果

10.8　设置手机的铃声

实例 126	以编程的方式设置手机中的铃声
源码路径	光盘:\daima\126
视频路径	光盘:\视频\126
实例必备	126.使用 AudioEffect 处理音效.pdf ① AudioEffect 基础 ② AudioEffect 中的嵌套类 ③ AudioEffect 中的常量 ④ AudioEffect 中的公有方法

10.8.1　实例说明

在手机中，铃声设置是一个十分重要的功能。在 Android 中，通过 RingtoneManager 类来专门控制各种铃声。例如，常见的来电铃声、闹钟铃声和一些警告、信息通知。RingtoneManager 类中的常用方法如下所示。

☑ getActualDefaultRingtoneUri()：获取指定类型的当前默认铃声。

☑ getCursor()：返回所有可用铃声的游标。

☑ getDefaultType()：获取指定 URL 默认的铃声类型。

☑ getDefaultUri()：返回指定类型默认铃声的 URL。

☑　getRingtoneUri()：返回指定位置铃声的 URL。
☑　getRingtonePosition()：获取指定铃声的位置。
☑　getValidRingtoneUri()：获取一个可用铃声的位置。
☑　isDefault()：获取指定 URL 是否为默认的铃声。
☑　setActualDefaultRingtoneUri()：设置默认的铃声。

在 Android 系统中，默认的铃声存储在 system\medio\audio 中，而下载的铃声一般保存在 SD 卡中，在此假设下载的铃声分别保存在 SD 卡的下述目录中。

☑　sdcard\music\ringtone：一般来电铃声。
☑　sdcard\music\alarm：闹钟铃声。
☑　sdcard\music\notification：警告、通知铃声。

10.8.2　具体实现

编写主程序文件，其具体实现流程如下所示。

（1）分别设置 3 个按钮对象、3 个自定义类型和 3 个铃声文件夹，具体代码如下所示。

```
/*3 个按钮*/
private Button mButtonRingtone;
private Button mButtonAlarm;
private Button mButtonNotification;
/*自定义的类型*/
public static final int ButtonRingtone = 0;
public static final int ButtonAlarm = 1;
public static final int ButtonNotification = 2;
/*铃声文件夹*/
private String strRingtoneFolder = "/sdcard/music/ringtone";
private String strAlarmFolder = "/sdcard/music/alarm";
private String strNotificationFolder = "/sdcard/music/notification";
```

（2）编写单击设置来电铃声按钮 mButtonRingtone 后的处理事件，先打开系统铃声设置，然后设置铃声，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mButtonRingtone = (Button) findViewById(R.id.ButtonRingtone);
    mButtonAlarm = (Button) findViewById(R.id.ButtonAlarm);
    mButtonNotification = (Button) findViewById(R.id.ButtonNotification);
    /*设置来电铃声*/
    mButtonRingtone.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View arg0)
        {
            if (bFolder(strRingtoneFolder))
            {
                //打开系统铃声设置
                Intent intent = new Intent(RingtoneManager.ACTION_RINGTONE_PICKER);
                //类型为来电 RINGTONE
```

```
                        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE,RingtoneManager. TYPE_
RINGTONE);
                        //设置显示的 title
                        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, "设置来电铃声");
                        //当设置完成之后返回到当前的 Activity
                        startActivityForResult(intent, ButtonRingtone);
                    }
                }
          });
```

（3）编写单击设置闹钟铃声按钮 mButtonAlarm 后的处理事件，先打开系统闹钟铃声设置，然后设置闹钟铃声，具体代码如下所示。

```
          /*设置闹钟铃声*/
          mButtonAlarm.setOnClickListener(new Button.OnClickListener()
          {
                @Override
                public void onClick(View arg0)
                {
                    if (bFolder(strAlarmFolder))
                    {
                        //打开系统铃声设置
                        Intent intent = new Intent(RingtoneManager.ACTION_RINGTONE_PICKER);
                        //设置铃声类型和 title
                        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE, RingtoneManager.TYPE_
ALARM);
                        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, "设置闹钟铃声");
                        //当设置完成之后返回到当前的 Activity
                        startActivityForResult(intent, ButtonAlarm);
                    }
                }
          });
```

（4）编写单击通知铃声按钮 mButtonNotification 的处理事件，先打开系统铃声设置，然后设置铃声，具体代码如下所示。

```
          /*设置通知铃声*/
          mButtonNotification.setOnClickListener(new Button.OnClickListener()
          {
                @Override
                public void onClick(View arg0)
                {
                    if (bFolder(strNotificationFolder))
                    {
                        //打开系统铃声设置
                        Intent intent = new Intent(RingtoneManager.ACTION_RINGTONE_PICKER);
                        //设置铃声类型和 title
                        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE, RingtoneManager.TYPE_
NOTIFICATION);
                        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, "设置通知铃声");
                        //当设置完成之后返回到当前的 Activity
                        startActivityForResult(intent, ButtonNotification);
                    }
                }
          });
```

（5）定义方法 bFolder()来检测是否存在指定的文件夹，如果不存在则创建一个，具体代码如下所示。

```
private boolean bFolder(String strFolder)
{
	boolean btmp = false;
	File f = new File(strFolder);
	if (!f.exists())
	{
		if (f.mkdirs())
		{
			btmp = true;
		}
		else
		{
			btmp = false;
		}
	}
	else
	{
		btmp = true;
	}
	return btmp;
}
```

执行代码后可以分别设置 3 种类型的铃声，效果如图 10-9 所示。



图 10-9　执行效果

10.9　播放远程网络中的 MP3

实例 127	播放远程网络中的 MP3
源码路径	光盘:\daima\127
视频路径	光盘:\视频\127
实例必备	127.语音识别技术.pdf ① Text-To-Speech 技术 ② 谷歌的 Voice Recognition 技术

10.9.1　实例说明

为了节约手机的存储空间，可以从网络中下载的方式播放 MP3。在本实例中，首先插入 4 个按钮，分别用于播放、暂停、重新播放和停止处理。执行后，通过 Runnable 发起运行线程，在线程中通过网

络传输方式远程下载指定的 MP3 文件。下载完毕后，临时保存到 SD 卡中，这样可以通过 4 个按钮对其进行控制。当程序关闭后，删除 SD 卡中的临时文件。

10.9.2 具体实现

编写主程序文件，具体实现流程如下所示。

（1）定义 currentFilePath 用于记录当前正在播放 MP3 的 URL 地址，定义 currentTempFilePath（当前播放 MP3 的路径），具体代码如下所示。

```
/*记录当前正在播放 MP3 的地址 URL*/
private String currentFilePath = "";
/*当前播放 MP3 的路径*/
private String currentTempFilePath = "";
private String strVideoURL = "";
```

（2）使用 strVideoURL 设置要播放 MP3 文件的网址，并设置透明度，具体代码如下所示。

```
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  /*MP3 文件不会被下载到 local*/
  strVideoURL = "http://www.lrn.cn/zywh/xyyy/yyxs/200805/W020080505536315331317.mp3";
  mTextView01 = (TextView)findViewById(R.id.myTextView1);
  /*设置透明度*/
  getWindow().setFormat(PixelFormat.TRANSPARENT);
  mPlay = (ImageButton)findViewById(R.id.play);
  mReset = (ImageButton)findViewById(R.id.reset);
  mPause = (ImageButton)findViewById(R.id.pause);
  mStop = (ImageButton)findViewById(R.id.stop);
```

（3）编写单击“播放”按钮所触发的处理事件，具体代码如下所示。

```
  /*开始播放*/
  mPlay.setOnClickListener(new ImageButton.OnClickListener()
  {
    public void onClick(View view)
    {
      /*调用播放影片 Function*/
      playVideo(strVideoURL);
      mTextView01.setText
      (
        getResources().getText(R.string.str_play).toString()+
        "\n"+ strVideoURL
      );
    }
  });
```

（4）编写单击“重播”按钮所触发的处理事件，具体代码如下所示。

```
  /*重新播放*/
  mReset.setOnClickListener(new ImageButton.OnClickListener()
  {
    public void onClick(View view)
    {
```

```
            if(bIsReleased == false)
            {
                if (mMediaPlayer01 != null)
                {
                    mMediaPlayer01.seekTo(0);
                    mTextView01.setText(R.string.str_play);
                }
            }
        }
    });
```

（5）编写单击“暂停”按钮所触发的处理事件，具体代码如下所示。

```
    /*暂停播放*/
    mPause.setOnClickListener(new ImageButton.OnClickListener()
    {
        public void onClick(View view)
        {
            if (mMediaPlayer01 != null)
            {
                if(bIsReleased == false)
                {
                    if(bIsPaused==false)
                    {
                        mMediaPlayer01.pause();
                        bIsPaused = true;
                        mTextView01.setText(R.string.str_pause);
                    }
                    else if(bIsPaused==true)
                    {
                        mMediaPlayer01.start();
                        bIsPaused = false;
                        mTextView01.setText(R.string.str_play);
                    }
                }
            }
        }
    });
```

（6）编写单击“停止”按钮所触发的处理事件，具体代码如下所示。

```
    /*停止播放*/
    mStop.setOnClickListener(new ImageButton.OnClickListener()
    {
        public void onClick(View view)
        {
            try
            {
                if (mMediaPlayer01 != null)
                {
                    if(bIsReleased==false)
                    {
                        mMediaPlayer01.seekTo(0);
                        mMediaPlayer01.pause();
```

```
                //mMediaPlayer01.stop();
                //mMediaPlayer01.release();
                //bIsReleased = true;
                mTextView01.setText(R.string.str_stop);
              }
            }
          }
          catch(Exception e)
          {
            mTextView01.setText(e.toString());
            Log.e(TAG, e.toString());
            e.printStackTrace();
          }
        }
      });
    }
```

（7）定义方法 playVideo(final String strPath)来播放指定的 MP3，播放的是存储卡中暂时保存的 MP3 文件，具体代码如下所示。

```
private void playVideo(final String strPath)
{
  try
  {
    if (strPath.equals(currentFilePath)&& mMediaPlayer01 != null)
    {
      mMediaPlayer01.start();
      return;
    }
    currentFilePath = strPath;
    mMediaPlayer01 = new MediaPlayer();
    mMediaPlayer01.setAudioStreamType(2);
```

（8）编写 setOnErrorListener()方法来监听错误处理，具体代码如下所示。

```
    /*错误事件*/
    mMediaPlayer01.setOnErrorListener(new MediaPlayer.OnErrorListener()
    {
      @Override
      public boolean onError(MediaPlayer mp, int what, int extra)
      {
        //TODO Auto-generated method stub
        Log.i(TAG, "Error on Listener, what: " + what + "extra: " + extra);
        return false;
      }
    });
```

（9）编写 setOnBufferingUpdateListener()方法来监听 MediaPlayer 缓冲区的更新，具体代码如下所示。

```
    /*捕捉使用 MediaPlayer 缓冲区的更新事件*/
    mMediaPlayer01.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener()
    {
      @Override
      public void onBufferingUpdate(MediaPlayer mp, int percent)
```

```
    {
        //TODO Auto-generated method stub
        Log.i(TAG, "Update buffer: " + Integer.toString(percent)+ "%");
    }
});
```

（10）编写 setOnCompletionListener()方法来监听播放完毕所触发的事件，具体代码如下所示。

```
/*播放完毕所触发的事件*/
mMediaPlayer01.setOnCompletionListener(new MediaPlayer.OnCompletionListener()
{
    @Override
    public void onCompletion(MediaPlayer mp)
    {
        //TODO Auto-generated method stub
        //delFile(currentTempFilePath);
        Log.i(TAG,"mMediaPlayer01 Listener Completed");
    }
});
```

（11）编写 setOnPreparedListener()方法来监听开始阶段的事件，具体代码如下所示。

```
/*开始阶段的监听 Listener*/
mMediaPlayer01.setOnPreparedListener(new MediaPlayer.OnPreparedListener()
{
    @Override
    public void onPrepared(MediaPlayer mp)
    {
        //TODO Auto-generated method stub
        Log.i(TAG,"Prepared Listener");
    }
});
```

（12）将文件保存到 SD 卡后，通过方法 mMediaPlayer01.start()播放 MP3，具体代码如下所示。

```
/*用 Runnable 来确保文件在存储完毕后才开始 start()*/
Runnable r = new Runnable()
{
    public void run()
    {
        try
        {
            /*setDataSource 将文件存到 SD 卡*/
            setDataSource(strPath);
            /*因为线程顺利进行，所以在 setDataSource 后运行 prepare()*/
            mMediaPlayer01.prepare();
            Log.i(TAG, "Duration: " + mMediaPlayer01.getDuration());

            /*开始播放 MP3*/
            mMediaPlayer01.start();
            bIsReleased = false;
        }
        catch (Exception e)
        {
            Log.e(TAG, e.getMessage(), e);
        }
```

```
            }
        };
        new Thread(r).start();
    }
```

（13）如果有异常则输出提示，具体代码如下所示。

```
    catch(Exception e)
    {
        if (mMediaPlayer01 != null)
        {
            /*线程发生异常则停止播放*/
            mMediaPlayer01.stop();
            mMediaPlayer01.release();
        }
        e.printStackTrace();
    }
```

（14）定义函数 setDataSource 用于存储 URL 的 MP3 文件到存储卡。首先判断传入的地址是否为 URL，然后创建 URL 对象和临时文件，具体实现代码如下所示。

```
/*定义函数用于存储 URL 的 MP3 文件到存储卡*/
private void setDataSource(String strPath) throws Exception
{
    /*判断传入的地址是否为 URL*/
    if (!URLUtil.isNetworkUrl(strPath))
    {
        mMediaPlayer01.setDataSource(strPath);
    }
    else
    {
        if(bIsReleased == false)
        {
            /*创建 URL 对象*/
            URL myURL = new URL(strPath);
            URLConnection conn = myURL.openConnection();
            conn.connect();

            /*获取 URLConnection 的 InputStream*/
            InputStream is = conn.getInputStream();
            if (is == null)
            {
                throw new RuntimeException("stream is null");
            }
            /*创建临时文件*/
            File myTempFile = File.createTempFile("yinyue", "."+getFileExtension(strPath));
            currentTempFilePath = myTempFile.getAbsolutePath();
            FileOutputStream fos = new FileOutputStream(myTempFile);
            byte buf[ ] = new byte[128];
            do
            {
                int numread = is.read(buf);
                if (numread <= 0)
                {
```

```
                break;
            }
            fos.write(buf, 0, numread);
        }while (true);

        /*直到 fos 存储完毕，调用 MediaPlayer.setDataSource*/
        mMediaPlayer01.setDataSource(currentTempFilePath);
        try
        {
            is.close();
        }
        catch (Exception ex)
        {
            Log.e(TAG, "error: " + ex.getMessage(), ex);
        }
    }
  }
}
```

（15）定义方法 getFileExtension(String strFileName)来获取音乐文件的扩展名，如果无法顺利获取扩展名，则默认为.dat，具体代码如下所示。

```
/*获取音乐文件扩展名自定义函数*/
private String getFileExtension(String strFileName)
{
    File myFile = new File(strFileName);
    String strFileExtension=myFile.getName();
    strFileExtension=(strFileExtension.substring(strFileExtension.lastIndexOf(".")+1)).toLowerCase();
    if(strFileExtension=="")
    {
        /*如果无法顺利获取扩展名则默认为.dat*/
        strFileExtension = "dat";
    }
    return strFileExtension;
}
```

（16）定义方法 delFile(String strFileName)来设置当退出程序时删除临时音乐文件，具体代码如下所示。

```
/*退出程序时需要调用自定义函数删除临时音乐文件*/
private void delFile(String strFileName)
{
    File myFile = new File(strFileName);
    if(myFile.exists())
    {
        myFile.delete();
    }
}

@Override
protected void onPause()
{
    //TODO Auto-generated method stub
```

```
    /*删除临时文件*/
    try
    {
      delFile(currentTempFilePath);
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    super.onPause();
  }
```

执行后可以通过“播放”“暂停”“重播”“停止”4 个按钮来控制指定的 MP3 音乐，如图 10-10 所示。

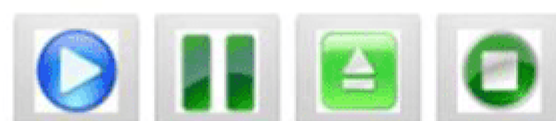


图 10-10　执行效果

第 11 章　手机游戏应用

自从手持设备诞生以来，游戏就成为最重要的应用之一。无论是在旅行和上下班的路上，或是闲暇时，都可以用手机游戏来打发无聊的时间。本章将通过几个典型实例的实现过程，详细介绍在 Android 系统中实现游戏项目的基本知识。

11.1　五子棋游戏

实例 128	五子棋游戏
源码路径	光盘:\daima\128
视频路径	光盘:\视频\128
实例必备	128.蓬勃发展的手机游戏产业.pdf ① 1.2 亿手机游戏用户 ② 淘金的时代 ③ 现实还需努力

11.1.1　实例说明

五子棋是一种两人对弈的纯策略型棋类游戏，起源于中国古代的传统黑白棋种之一。其发展于日本，流行于欧美，简单易学，老少皆宜，而且趣味横生，引人入胜，不仅能增强思维能力，提高智力，而且富含哲理，有助于修身养性。

许多国家的玩家对五子棋有不同的爱称，例如，韩国人把五子棋称为“情侣棋”，暗示情人之间下五子棋有利于增加情感的交流；欧洲人称其为“绅士棋”，代表下五子棋的君子风度胜似绅士；日本人则称其为“中老年棋”，说明五子棋适合中老年人的生理特点和思维方式；美国人喜欢将五子棋称为“商业棋”，也就是说，商人可边下棋边谈生意，棋下完了生意也谈成了。

11.1.2　具体实现

本实例实现了一个简单的五子棋功能，主界面有 3 个按钮，分别是“重玩”“选项”“退出”。界面大部分都是方格棋盘，在上面可以摆放五子棋的黑棋子和白棋子。本实例比较简单，具体实现流程如下所示。

（1）编写布局文件 main.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

（2）编写文件 Const.java，这是一个常量文件，将系统中需要的量在此文件中统一定义，这样做的好处是方便对系统的维护和理解，主要代码如下所示。

```
public interface Const {
    public static final int ALIGN_TOP = 1;
    public static final int ALIGN_VCENTER = ALIGN_TOP << 1;
    public static final int ALIGN_LEFT = ALIGN_TOP << 2;
    public static final int ALIGN_RIGHT = ALIGN_TOP << 3;
    public static final int ALIGN_HCENTER = ALIGN_TOP << 4;
    public static final int ALIGN_BOTTOM = ALIGN_TOP << 5;
    public final static int GS_WAIT = 0;
    public final static int GS_INVITING = 1;
    public final static int GS_COMFIRE = 2;
    public final static int GS_DECLINE = 3;
    public final static int GS_GAME = 4;
    public final static int GS_END = 5;
    public final static int GS_AWAY = 6;
    public final static int GS_ERROR = 7;
    public final static int MAP_SPACE = 15;
    public final static int TILE_WIDTH = 24;
    public final static int TILE_HEIGHT = 25;
    public final static int CHESS_WIDTH = 9;
    public final static int CHESS_HEIGHT = 9;
    public final static int RADIUS_SPACE = TILE_WIDTH >>1;
    public final static int CAMP_DEFAULT = 0;
    public final static int CAMP_HERO = 1;
    public final static int CAMP_ENEMY = 2;
    public final static int CALU_ALL_COUNT = 10;
    public final static int CALU_SINGLE_COUNT = 5;
}
```

（3）编写文件 mainA.java，此文件实现游戏主界面，能够实现标题栏隐藏功能和全屏显示功能，主要代码如下所示。

```
public class mainA extends Activity {
    GameV gameView = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     //隐藏标题栏
     requestWindowFeature(Window.FEATURE_NO_TITLE);
     //全屏显示
     getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

```
        //获取屏幕宽高
        Display display = getWindowManager().getDefaultDisplay();
        //现实 GameView
        GameV.init(this, display.getWidth(), display.getHeight());
        gameView = GameV.getInstance();
        setContentView(gameView);
       }
       public boolean onKeyDown(int keyCode, KeyEvent event) {
        return super.onKeyDown(keyCode, event);
       }
}
```

（4）编写文件 GameV.java，此文件是该五子棋游戏的核心，在其中定义了继承于 SurfaceView 类的子类 GameV，该类实现了整个游戏框架功能。文件 GameV.java 的具体实现流程如下所示。

① 定义了继承于 SurfaceView 类的子类 GameV，定义了游戏框架界面中元素的初始值，主要代码如下所示。

```
public class GameV extends SurfaceView implements Const,
        SurfaceHolder.Callback, Runnable {
       static GameV sInstance = null;
       public static void init(Activity mActivity, int screenWidth,
               int screenHeight) {
        sInstance = new GameV(mActivity, screenWidth, screenHeight);
       }
       public static GameV getInstance() {
        return sInstance;
       }
       //控制循环
       boolean mbLoop = false;
       //定义 SurfaceHolder 对象
       SurfaceHolder mSurfaceHolder = null;
       public static Paint sPaint = null;
       public static Canvas sCanvas = null;
       public static Resources sResources = null;
       private int mGameState = 0;
       private int mScreenWidth = 0;
       private int mScreenHeight = 0;
       public int[ ][ ] mGameMap = null;
       private int mMapHeightLengh = 0;
       private int mMapWidthLengh = 0;

       private int mMapIndexX = 0;
       private int mMapIndexY = 0;
       public int mCampTurn = 0;
       public int mCampWinner = 0;
       private float mTitleSpace = 0;
       private int mTitleHeight = 0;
       private float mTitleIndex_x = 0;
       private float mTitleIndex_y = 0;
       Bitmap bitmapBg = null;
       Bitmap mBlack = null;
       Bitmap mWhite = null;
```

```
Context mContext = null;
public GameV(Activity activity, int screenWidth, int screenHeight) {
 super(activity);
 sPaint = new Paint();
 sPaint.setAntiAlias(true);
 sResources = getResources();
 mContext = activity;
 mScreenWidth = screenWidth;
 mScreenHeight = screenHeight;
 mSurfaceHolder = this.getHolder();
 mSurfaceHolder.addCallback(this);
 setFocusable(true);
 mbLoop = true;
 bitmapBg = CreatMatrixBitmap(R.drawable.status, mScreenWidth,
         mScreenHeight);
 mBlack = BitmapFactory.decodeResource(GameV.sResources,
         R.drawable.ai);
 mWhite = BitmapFactory.decodeResource(GameV.sResources,
         R.drawable.human);
 mTitleSpace = (float) mScreenWidth / CHESS_WIDTH;
 mTitleHeight = mScreenHeight / 3;
 mTitleIndex_x = (float) (mTitleSpace / 2);
 mTitleIndex_y = (float) (mTitleSpace / 2);
 setGameState(GS_GAME);
}
```

② 定义方法 onTouchEvent()，功能是根据用户触摸屏幕实现走棋响应，具体代码如下所示。

```
public boolean onTouchEvent(MotionEvent event) {
 int x = (int) event.getX();
 int y = (int) event.getY();
 switch (event.getAction()) {
 case MotionEvent.ACTION_DOWN:
       UpdateTouchEvent(x, y);
       break;
 case MotionEvent.ACTION_MOVE:
       break;
 case MotionEvent.ACTION_UP:
       break;
 }
 return super.onTouchEvent(event);
}

public boolean CheckPiecesMeet(int Camp) {
 int MeetCount = 0;
 //横向
 for (int i = 0; i < CALU_ALL_COUNT; i++) {
       int index = mMapIndexX - CALU_SINGLE_COUNT + i;
       if (index < 0 || index >= mMapWidthLengh) {
        if (MeetCount == CALU_SINGLE_COUNT) {
              return true;
        }
```

```
            MeetCount = 0;
            continue;
        }
        if (mGameMap[mMapIndexY][index] == Camp) {
            MeetCount++;
            if (MeetCount == CALU_SINGLE_COUNT) {
                return true;
            }
        } else {
            MeetCount = 0;
        }
}
//纵向
MeetCount = 0;
for (int i = 0; i < CALU_ALL_COUNT; i++) {
        int index = mMapIndexY - CALU_SINGLE_COUNT + i;
        if (index < 0 || index >= mMapHeightLengh) {
            if (MeetCount == CALU_SINGLE_COUNT) {
                return true;
            }
            MeetCount = 0;
            continue;
        }
        if (mGameMap[index][mMapIndexX] == Camp) {
            MeetCount++;
            if (MeetCount == CALU_SINGLE_COUNT) {
                return true;
            }
        } else {
            MeetCount = 0;
        }
}

//右斜
MeetCount = 0;
for (int i = 0; i < CALU_ALL_COUNT; i++) {
        int indexX = mMapIndexX - CALU_SINGLE_COUNT + i;
        int indexY = mMapIndexY - CALU_SINGLE_COUNT + i;
        if ((indexX < 0 || indexX >= mMapWidthLengh)
                || (indexY < 0 || indexY >= mMapHeightLengh)) {
            if (MeetCount == CALU_SINGLE_COUNT) {
                return true;
            }
            MeetCount = 0;
            continue;
        }
        if (mGameMap[indexY][indexX] == Camp) {
            MeetCount++;
            if (MeetCount == CALU_SINGLE_COUNT) {
                return true;
```

```
            }
          } else {
            MeetCount = 0;
          }
      }

      //左斜
      MeetCount = 0;
      for (int i = 0; i < CALU_ALL_COUNT; i++) {
            int indexX = mMapIndexX - CALU_SINGLE_COUNT + i;
            int indexY = mMapIndexY + CALU_SINGLE_COUNT - i;
            if ((indexX < 0 || indexX >= mMapWidthLengh)
                  || (indexY < 0 || indexY >= mMapHeightLengh)) {
             if (MeetCount == CALU_SINGLE_COUNT) {
                  return true;
             }
             MeetCount = 0;
             continue;
            }
            if (mGameMap[indexY][indexX] == Camp) {
             MeetCount++;
             if (MeetCount == CALU_SINGLE_COUNT) {
                  return true;
             }
            } else {
             MeetCount = 0;
            }
      }
      return false;
     }
```

③ 定义方法 UpdateTouchEvent()，功能是当触摸屏幕棋盘时实现屏幕内容的更新，从而实现下棋功能，具体代码如下所示。

```
     private void UpdateTouchEvent(int x, int y) {
      switch (mGameState) {
      case GS_GAME:
            if (x > 0 && y > mTitleHeight) {
             mMapIndexX = (int) (x / mTitleSpace);
             mMapIndexY = (int) ((y - mTitleHeight) / mTitleSpace);

             if (mMapIndexX > mMapWidthLengh) {
                  mMapIndexX = mMapWidthLengh;
             }
             if (mMapIndexX < 0) {
                  mMapIndexX = 0;
             }

             if (mMapIndexY > mMapHeightLengh) {
                  mMapIndexY = mMapHeightLengh;
             }
```

```
            if (mMapIndexY < 0) {
                mMapIndexY = 0;
            }
            if (mGameMap[mMapIndexY][mMapIndexX] == CAMP_DEFAULT) {

                if (mCampTurn == CAMP_HERO) {
                 mGameMap[mMapIndexY][mMapIndexX] = CAMP_HERO;
                 if (CheckPiecesMeet(CAMP_HERO)) {
                     mCampWinner = R.string.Role_black;
                     setGameState(GS_END);
                 } else {
                     mCampTurn = CAMP_ENEMY;
                 }

                } else {
                 mGameMap[mMapIndexY][mMapIndexX] = CAMP_ENEMY;
                 if (CheckPiecesMeet(CAMP_ENEMY)) {
                     mCampWinner = R.string.Role_white;
                     setGameState(GS_END);
                 } else {
                     mCampTurn = CAMP_HERO;
                 }
                }
            }
           }
           break;
       case GS_END:
           setGameState(GS_GAME);
           break;

       }
      }
```

④ 定义方法 CreatMatrixBitmap()，功能是创建一个缩小或放大的新图片，具体代码如下所示。

```
      private Bitmap CreatMatrixBitmap(int resourcesID, float scr_width,
           float res_height) {
       Bitmap bitMap = null;
       bitMap = BitmapFactory.decodeResource(sResources, resourcesID);
       int bitWidth = bitMap.getWidth();
       int bitHeight = bitMap.getHeight();
       float scaleWidth = scr_width / (float) bitWidth;
       float scaleHeight = res_height / (float) bitHeight;
       Matrix matrix = new Matrix();
       matrix.postScale(scaleWidth, scaleHeight);
       bitMap = Bitmap.createBitmap(bitMap, 0, 0, bitWidth, bitHeight, matrix,
            true);
       return bitMap;
      }
```

⑤ 定义方法 DrawString()，功能是在屏幕中绘制一个字符串，具体代码如下所示。

```
      private void DrawString(int color, String text, int x, int y, int anchor) {
       Rect rect = new Rect();
```

```
        sPaint.getTextBounds(text, 0, text.length(), rect);
        int w = rect.width();
        int h = rect.height();
        int tx = 0;
        int ty = 0;
        if ((anchor & ALIGN_RIGHT) != 0) {
            tx = x - w;
        } else if ((anchor & ALIGN_HCENTER) != 0) {
            tx = x - (w >> 1);
        } else {
            tx = x;
        }
        if ((anchor & ALIGN_TOP) != 0) {
            ty = y + h;
        } else if ((anchor & ALIGN_VCENTER) != 0) {
            ty = y + (h >> 1);
        } else {
            ty = y;
        }
        sPaint.setColor(color);
        sCanvas.drawText(text, tx, ty, sPaint);
    }
```

⑥ 定义方法 DrawImage()，功能是绘制一张图片，可以选择图片的锚点位置。在此有 3 个锚点位置，分别是“重玩”“选项”“退出”，具体代码如下所示。

```
    private void DrawImage(Bitmap bitmap, float x, float y, int anchor) {
        int w = bitmap.getWidth();
        int h = bitmap.getHeight();
        float tx = 0;
        float ty = 0;
        if ((anchor & ALIGN_RIGHT) != 0) {
            tx = x - w;
        } else if ((anchor & ALIGN_HCENTER) != 0) {
            tx = x - (w >> 1);
        } else {
            tx = x;
        }
        if ((anchor & ALIGN_TOP) != 0) {
            ty = y + h;
        } else if ((anchor & ALIGN_VCENTER) != 0) {
            ty = y - (h >> 1);
        } else if ((anchor & ALIGN_BOTTOM) != 0) {
            ty = y - h;
        } else {
            ty = y;
        }
        sCanvas.drawBitmap(bitmap, tx, ty, sPaint);
    }
```

至此，本五子棋游戏介绍完毕，执行后的效果如图 11-1 所示。



图 11-1　执行效果

11.2　益智类游戏——魔塔

实例 129	益智类游戏——魔塔
源码路径	光盘:\daima\129
视频路径	光盘:\视频\129
实例必备	129.游戏开发流程.pdf

11.2.1　实例说明

魔塔是一种固定数值的 RPG 游戏，在玩此游戏时需要动很多脑筋，任何一个轻率的选择都可能导致游戏失败。此游戏还可以锻炼游戏者的数学能力。本实例基于 Android 平台，开发了一个经典魔塔游戏。

11.2.2　具体实现

1．设计游戏框架

因为所有游戏是基于框架的，所以设计一个合理、科学的框架尤为重要。为了使框架更完美，先看市面上魔塔游戏的界面，吸取其中精华，学习经验，如图 11-2 所示。

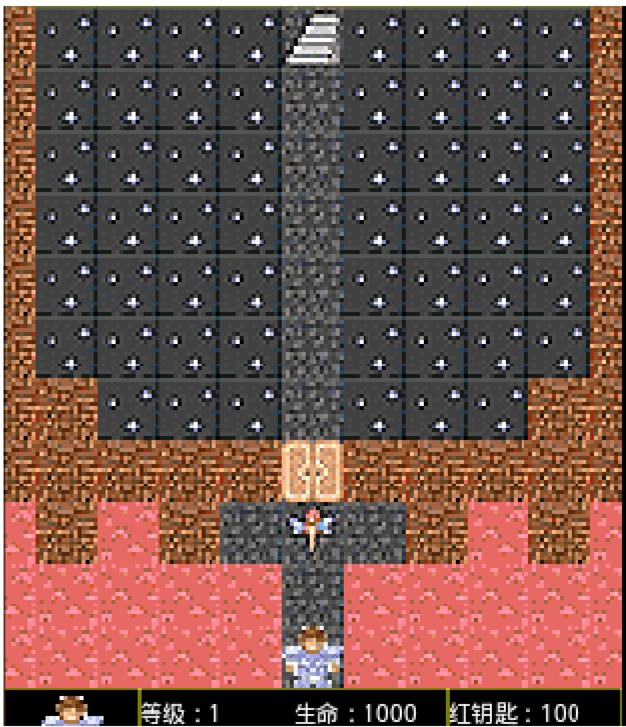


图 11-2　市面魔塔游戏界面

由图 11-2 所示游戏界面可知，在游戏中包含了地图、角色、屏幕界面、道具等元素，这些元素构成了一个视图，如屏幕视图、道具视图、角色视图等。

（1）界面视图。在 Android 中，视图是通过继承 View 类实现的，在 View 类中包含了各种绘制图形的方法和事件处理，如 onKeyDown、onKeyUp 等。在构造此视图类时还可以加入自己的一些抽象方法，如资源回收和刷新等。界面类 GameView 类的主要实现代码如下所示。

```
public abstract class GameView extends View
{
```

```
    public GameView(Context context)
    {
        super(context);
    }
    /*绘图*/
    protected abstract void onDraw(Canvas canvas);
    /*按键按下*/
    public abstract boolean onKeyDown(int keyCode);
    /*按键弹起*/
    public abstract boolean onKeyUp(int keyCode);
    /*回收资源*/
    protected abstract void reCycle();

    /*刷新*/
    protected abstract void refurbish();
}
```

（2）屏幕显示。前面的视图类用于显示游戏界面，此外，还需要控制当前的屏幕显示哪一个界面，并且能够对界面进行一些逻辑上的处理，这就需要建立一个整个游戏的 MainGame 类，在此类中需要根据不同的游戏状态来设置屏幕需要显示的视图。在此需要编写 MainGame 类，具体代码如下所示。

```
public class MainGame
{
    private static GameView m_GameView = null;        //当前需要显示的对象
    private Context m_Context = null;
    private MagicTower m_MagicTower  = null;
    private int m_status = -1;                         //游戏状态
    public CMIDIPlayer mCMIDIPlayer;
    public byte mbMusic = 0;
    public MainGame(Context context)
    {
        m_Context = context;
        m_MagicTower = (MagicTower)context;
        m_status = -1;

        initGame();
    }
    //初始化游戏
    public void initGame()
    {
        controlView(yarin.GAME_SPLASH);
        mCMIDIPlayer = new CMIDIPlayer(m_MagicTower);
    }
    //得到游戏状态
    public int getStatus()
    {
        return m_status;
    }
    //设置游戏状态
    public void setStatus(int status)
    {
        m_status = status;
```

```
}
//得到主类对象
public Activity getMagicTower()
{
        return m_MagicTower;
}

//得到当前需要显示的对象
public static GameView getMainView()
{
       return m_GameView;
}
//控制显示的界面
public void controlView(int status)
{
        if(m_status != status)
        {
                if(m_GameView != null)
                {
                        m_GameView.reCycle();
                        System.gc();
                }
        }
        freeGameView(m_GameView);
        switch (status)
        {
        case yarin.GAME_SPLASH:
                m_GameView = new SplashScreen(m_Context,this);
                break;
        case yarin.GAME_MENU:
                m_GameView = new MainMenu(m_Context,this);
                break;
        case yarin.GAME_HELP:
                m_GameView = new HelpScreen(m_Context,this);
                break;
        case yarin.GAME_ABOUT:
                m_GameView = new AboutScreen(m_Context,this);
                break;
        case yarin.GAME_RUN:
                m_GameView = new GameScreen(m_Context,m_MagicTower,this,true);
                break;
        case yarin.GAME_CONTINUE:
                m_GameView = new GameScreen(m_Context,m_MagicTower,this,false);
                break;
        }
        setStatus(status);
}

//释放界面对象
public void freeGameView(GameView gameView)
{
```

```
            if(gameView != null)
            {
                gameView = null;
                System.gc();
            }
        }
}
```

（3）更新线程。当创建和控制视图显示以后，还需要让游戏能够动起来，此时需要线程来实现界面的自动更新。在此可以为游戏开启一个主线程，并通过方法 MainGame.getMainView()来获取当前显示的界面，并根据不同的界面进行游戏更新。此功能在文件 ThreadCanvas.java 中实现，主要代码如下所示。

```
package com.example205;
public class ThreadCanvas extends View implements Runnable
{
    private String   m_Tag = "ThreadCanvas_Tag";
    public ThreadCanvas(Context context)
    {
        super(context);
    }
    /*绘图*/
    protected void onDraw(Canvas canvas)
    {
        if (MainGame.getMainView() != null)
        {
            MainGame.getMainView().onDraw(canvas);
        }
        else
        {
            Log.i(m_Tag, "null");
        }
    }
    /*绘图显示*/
    public void start()
    {
        Thread t = new Thread(this);
        t.start();
    }
    //刷新界面
    public void refurbish()
    {
        if (MainGame.getMainView() != null)
        {
            MainGame.getMainView().refurbish();
        }
    }

    /*游戏循环*/
    public void run()
    {
        while (true)
```

```
        {
            try
            {
                Thread.sleep(yarin.GAME_LOOP);
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
            refurbish();                            //更新显示
            postInvalidate();                       //刷新屏幕
        }
    }
    //按键处理（按键按下）
    boolean onKeyDown(int keyCode)
    {
        if (MainGame.getMainView() != null)
        {
            MainGame.getMainView().onKeyDown(keyCode);
        }
        else
        {
            Log.i(m_Tag, "null");
        }
        return true;
    }
    //按键弹起
    boolean onKeyUp(int keyCode)
    {
        if (MainGame.getMainView() != null)
        {
            MainGame.getMainView().onKeyUp(keyCode);
        }
        else
        {
            Log.i(m_Tag, "null");
        }
        return true;
    }
}
```

（4）显示。接下来需要调用 Activity 来显示具体的界面，因为是在 ThreadCanvas 中控制界面的，所以需要用 setContentView()方法来显示一个 ThreadCanvas 对象。本实例的显示功能通过文件 MagicTower.java 实现，主要代码如下所示。

```
public class MagicTower extends Activity
{
    private ThreadCanvas mThreadCanvas = null;
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setTheme(android.R.style.Theme_Black_NoTitleBar_Fullscreen);
```

```
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.Layout Params.
FLAG_FULLSCREEN);
            new MainGame(this);
            mThreadCanvas = new ThreadCanvas(this);
            setContentView(mThreadCanvas);
      }

      /*暂停*/
      protected void onPause()
      {
            super.onPause();
      }
      /*重绘*/
      protected void onResume()
      {
            super.onResume();
            mThreadCanvas.requestFocus();
            mThreadCanvas.start();
      }
      /*按键按下*/
      public boolean onKeyDown(int keyCode, KeyEvent event)
      {
            mThreadCanvas.onKeyDown(keyCode);
            return false;
      }
      /*按键弹起*/
      public boolean onKeyUp(int keyCode, KeyEvent event)
      {
            mThreadCanvas.onKeyUp(keyCode);
            return false;
      }
}
```

2．后面的视图

游戏框架设计的完成，标志着整个游戏项目的根基已经打好了。接下来需要在此基础上进一步完善，此魔塔游戏后面的功能也都是基于视图的，下面详细介绍具体实现过程。

（1）设计地图。地图在游戏项目中十分重要，游戏中的所有角色都需要在某个场景中生存。但是不可能让美工制作一张巨大的地图，因为这样的游戏将会很大，不利于在手机有限的空间和配置中使用。通常游戏中的地图是由多个小块构成的一个完整大图，所以完全可以使用一个二维数组来存储小块数据，然后通过程序将地图数据对应的小块映射到屏幕，从而组成一幅完整的地图。当前主流的做法是用 mappy 来生成地图，然后用脚本语言为 mappy 写一个保存格式的程序。一般情况下，地图分为 45 度角、仰视角和俯视角。

实现地图的思路非常清晰，具体流程如下所示。

① 创建一个对象类，在此命名为 TiledLayer，具体代码如下所示。

```
public class TiledLayer extends Layer
```

② 新建对应的构造函数，具体代码如下所示。

```
public TiledLayer(int columns, int rows, Bitmap image, int tileWidth,
        int tileHeight) {
    super(columns < 1 || tileWidth < 1 ? -1 : columns * tileWidth, rows < 1
            || tileHeight < 1 ? -1 : rows * tileHeight);
    if (((image.getWidth() % tileWidth) != 0)
            || ((image.getHeight() % tileHeight) != 0)) {
        throw new IllegalArgumentException();
    }
    this.columns = columns;
    this.rows = rows;
    cellMatrix = new int[rows][columns];
    int noOfFrames = (image.getWidth() / tileWidth)
            * (image.getHeight() / tileHeight);
    createStaticSet(image, noOfFrames + 1, tileWidth, tileHeight, true);
}
```

其中参数 columns 和 rows 分别表示地图的行数和列数，image 代表地图中的一块，tileWidth 和 tileHeight 分别表示图块的宽度和高度。

③ 定义方法 setCell(int col, int row, int tileIndex)来设置地图使用的数据，此方法的具体实现代码如下所示。

```
public void setCell(int col, int row, int tileIndex) {
    if (col < 0 || col >= this.columns || row < 0 || row >= this.rows) {
        throw new IndexOutOfBoundsException();
    }
    if (tileIndex > 0) {
        if (tileIndex >= numberOfTiles) {
            throw new IndexOutOfBoundsException();
        }
    } else if (tileIndex < 0) {
        // do animated tile index check
        if (anim_to_static == null || (-tileIndex) >= numOfAnimTiles) {
            throw new IndexOutOfBoundsException();
        }
    }
    cellMatrix[row][col] = tileIndex;
}
```

④ 通过方法 createAnimatedTile(int staticTileIndex)实现动态贴图功能，此方法在 J2ME 中十分常见。动态贴图可以通过调用这个方法来创建，该方法返回一个索引号，用于标记新创建的动态贴图。动态贴图的索引号总是负数，并且也是连续的，起始值为-1。一旦被创建，与之关联的静态贴图可以通过调用 setAnimatedTile(int, int)方法来改变，具体代码如下所示。

```
public int createAnimatedTile(int staticTileIndex) {
    if (staticTileIndex < 0 || staticTileIndex >= numberOfTiles) {
        throw new IndexOutOfBoundsException();
    }
    if (anim_to_static == null) {
        anim_to_static = new int[4];
        numOfAnimTiles = 1;
    } else if (numOfAnimTiles == anim_to_static.length) {
        // grow anim_to_static table if needed
        int new_anim_tbl[ ] = new int[anim_to_static.length * 2];
```

```
                System.arraycopy(anim_to_static, 0, new_anim_tbl, 0,
                        anim_to_static.length);
                anim_to_static = new_anim_tbl;
            }
            anim_to_static[numOfAnimTiles] = staticTileIndex;
            numOfAnimTiles++;
            return (-(numOfAnimTiles - 1));
        }
```

⑤ 定义方法 setAnimatedTile(int animatedTileIndex, int staticTileIndex)实现动态图块的内容，其第一个参数是动态图块的编号，第二个参数是 Tile 的编号，具体代码如下所示。

```
            public void setAnimatedTile(int animatedTileIndex, int staticTileIndex) {
            if (staticTileIndex < 0 || staticTileIndex >= numberOfTiles) {
                throw new IndexOutOfBoundsException();
            }
                    animatedTileIndex = -animatedTileIndex;
            if (anim_to_static == null || animatedTileIndex <= 0
                    || animatedTileIndex >= numOfAnimTiles) {
                throw new IndexOutOfBoundsException();
            }
            anim_to_static[animatedTileIndex] = staticTileIndex;
        }
```

⑥ 使用函数 getAnimatedTile(int animatedTileIndex)得到序号为 animatedTileIndex 的动画分块实际的图像分块序号，具体代码如下所示。

```
        public int getAnimatedTile(int animatedTileIndex) {
            animatedTileIndex = -animatedTileIndex;
            if (anim_to_static == null || animatedTileIndex <= 0
                    || animatedTileIndex >= numOfAnimTiles) {
                throw new IndexOutOfBoundsException();
            }
            return anim_to_static[animatedTileIndex];
        }
```

⑦ 使用方法 paint()将图绘制在屏幕上，具体代码如下所示。

```
        public void fillCells(int col, int row, int numCols, int numRows,
                int tileIndex) {
            if (col < 0 || col >= this.columns || row < 0 || row >= this.rows
                    || numCols < 0 || col + numCols > this.columns || numRows < 0
                    || row + numRows > this.rows) {
                throw new IndexOutOfBoundsException();
            }
            if (tileIndex > 0) {
                if (tileIndex >= numberOfTiles) {
                    throw new IndexOutOfBoundsException();
                }
            } else if (tileIndex < 0) {
                if (anim_to_static == null || (-tileIndex) >= numOfAnimTiles) {
                    throw new IndexOutOfBoundsException();
                }
            }
            for (int rowCount = row; rowCount < row + numRows; rowCount++) {
```

```
                for (int columnCount = col; columnCount < col + numCols; columnCount++) {
                    cellMatrix[rowCount][columnCount] = tileIndex;
                }
            }
        }
```

（2）设计主角。在本实例游戏中只有一个主角，在魔塔游戏中称为“精灵”，并且这个“精灵”角色还有很多动作。例如，向 4 个方向的移动，在移动时就是一个动画。动画本身是将图片一帧一帧连接起来、循环播放每一帧所形成的。在一些大型游戏中，可以使用精灵编辑器编写精灵，将精灵拆解为很多部分，然后再组合起来，这样可以节省大量的空间。此处使用 Sprite 类实现魔塔中的主角，该类是一个用于显示图像的类。下面讲解主角的具体实现过程。

① 构建 Sprite 对象。先定义 Sprite 类，代码如下所示。

```
public class Sprite extends Layer
```

然后在 Sprite 类中提供了如下 3 个构造方法来构建完整的 Sprite 类。

☑ 方法 Sprite(Bitmap image)的主要代码如下所示。

```
    public Sprite(Bitmap image) {
        super(image.getWidth(), image.getHeight());
        initializeFrames(image, image.getWidth(), image.getHeight(), false);
        initCollisionRectBounds();
        this.setTransformImpl(TRANS_NONE);
    }
```

☑ 方法 Sprite(Bitmap image, int frameWidth, int frameHeight)的具体实现代码如下所示。

```
    public Sprite(Bitmap image, int frameWidth, int frameHeight) {
        super(frameWidth, frameHeight);
        if ((frameWidth < 1 || frameHeight < 1)
                || ((image.getWidth() % frameWidth) != 0)
                || ((image.getHeight() % frameHeight) != 0)) {
            throw new IllegalArgumentException();
        }
        initializeFrames(image, frameWidth, frameHeight, false);
        initCollisionRectBounds();
        this.setTransformImpl(TRANS_NONE);
    }
```

☑ 方法 Sprite(Sprite s)的具体实现代码如下所示。

```
    public Sprite(Sprite s) {
        super(s != null ? s.getWidth() : 0, s != null ? s.getHeight() : 0);
        if (s == null) {
            throw new NullPointerException();
        }
        this.sourceImage = s.sourceImage;
        this.numberFrames = s.numberFrames;
        this.frameCoordsX = new int[this.numberFrames];
        this.frameCoordsY = new int[this.numberFrames];
        System.arraycopy(s.frameCoordsX, 0, this.frameCoordsX, 0, s
                .getRawFrameCount());
        System.arraycopy(s.frameCoordsY, 0, this.frameCoordsY, 0, s
                .getRawFrameCount());
        this.x = s.getX();
        this.y = s.getY();
```

```
            this.dRefX = s.dRefX;
            this.dRefY = s.dRefY;
            this.collisionRectX = s.collisionRectX;
            this.collisionRectY = s.collisionRectY;
            this.collisionRectWidth = s.collisionRectWidth;
            this.collisionRectHeight = s.collisionRectHeight;
            this.srcFrameWidth = s.srcFrameWidth;
            this.srcFrameHeight = s.srcFrameHeight;
            setTransformImpl(s.t_currentTransformation);
            this.setVisible(s.isVisible());
            this.frameSequence = new int[s.getFrameSequenceLength()];
            this.setFrameSequence(s.frameSequence);
            this.setFrame(s.getFrame());

            this.setRefPixelPosition(s.getRefPixelX(), s.getRefPixelY());
      }
```

在上述 3 个构造方法中，参数 image 为精灵的图片，参数 frameWidth 和 frameHeight 分别为精灵图片每一帧的宽度和高度，参数 s 表示通过一个精灵来创建另一个精灵。构建 Sprite 类时需要指定精灵的高度和宽度（像素值）。图像的高度和宽度必须分别是精灵的高度和宽度的整数倍，即能正好把图像按照精灵的大小划分成几个类。通过上面的例子，将帧在视图中排列成一个方阵，帧的排列没有严格限制，既有横排的，也有竖排的。接着就可以指定帧数了，左上方是编号 0，然后从左到右、从上到下依次排列。可以使用 setFrame(int sequenceIndex)选择哪一帧被显示，只要将其编号作为参数传递即可。

② 设置 Sprite 属性。因为类 TiledLayer 可以自动根据精灵的位置来判断地图绘制的位置，所以可以使用一个方法来设置精灵的位置。在此定义为 setRefPixelPosition(int x, int y)方法，具体代码如下所示。

```
      public void setRefPixelPosition(int x, int y) {
            // update this.x and this.y
            this.x = x
                        - getTransformedPtX(dRefX, dRefY, this.t_currentTransformation);
            this.y = y
                        - getTransformedPtY(dRefX, dRefY, this.t_currentTransformation);
      }
```

其中参数 x 和 y 是精灵的位置。

③ 实现碰撞检测处理。碰撞即相遇，当精灵和外物相碰撞时就需要对应的处理。本实例中精灵类提供了以下 3 个碰撞检测函数。

- ☑ public final boolean collidesWith(TiledLayer t, boolean pixelLevel)
- ☑ public final boolean collidesWith(Sprite s, boolean pixelLevel)
- ☑ public final boolean collidesWith(Bitmap image, int x, int y,boolean pixelLevel)

使用函数 collidesWith(TiledLayer t, boolean pixelLevel)实现精灵和 TiledLayer 的碰撞，主要代码如下所示。

```
      public final boolean collidesWith(TiledLayer t, boolean pixelLevel) {
            if (!(t.visible && this.visible)) {
                  return false;
            }
            int tLx1 = t.x;
            int tLy1 = t.y;
            int tLx2 = tLx1 + t.width;
```

```
int tLy2 = tLy1 + t.height;
int tW = t.getCellWidth();
int tH = t.getCellHeight();
int sx1 = this.x + this.t_collisionRectX;
int sy1 = this.y + this.t_collisionRectY;
int sx2 = sx1 + this.t_collisionRectWidth;
int sy2 = sy1 + this.t_collisionRectHeight;
int tNumCols = t.getColumns();
int tNumRows = t.getRows();
int startCol; // = 0;
int endCol; // = 0;
int startRow; // = 0;
int endRow; // = 0;
if (!intersectRect(tLx1, tLy1, tLx2, tLy2, sx1, sy1, sx2, sy2)) {
        return false;
}
startCol = (sx1 <= tLx1) ? 0 : (sx1 - tLx1) / tW;
startRow = (sy1 <= tLy1) ? 0 : (sy1 - tLy1) / tH;
endCol = (sx2 < tLx2) ? ((sx2 - 1 - tLx1) / tW) : tNumCols - 1;
endRow = (sy2 < tLy2) ? ((sy2 - 1 - tLy1) / tH) : tNumRows - 1;
if (!pixelLevel) {
                        for (int row = startRow; row <= endRow; row++) {
                for (int col = startCol; col <= endCol; col++) {
                        if (t.getCell(col, row) != 0) {
                                return true;
                        }
                }
        }
        return false;
} else {
        if (this.t_collisionRectX < 0) {
                sx1 = this.x;
        }
        if (this.t_collisionRectY < 0) {
                sy1 = this.y;
        }
        if ((this.t_collisionRectX + this.t_collisionRectWidth) > this.width) {
                sx2 = this.x + this.width;
        }
        if ((this.t_collisionRectY + this.t_collisionRectHeight) > this.height) {
                sy2 = this.y + this.height;
        }
        if (!intersectRect(tLx1, tLy1, tLx2, tLy2, sx1, sy1, sx2, sy2)) {
                return (false);
        }
        startCol = (sx1 <= tLx1) ? 0 : (sx1 - tLx1) / tW;
        startRow = (sy1 <= tLy1) ? 0 : (sy1 - tLy1) / tH;
        endCol = (sx2 < tLx2) ? ((sx2 - 1 - tLx1) / tW) : tNumCols - 1;
        endRow = (sy2 < tLy2) ? ((sy2 - 1 - tLy1) / tH) : tNumRows - 1;
        int cellTop = startRow * tH + tLy1;
        int cellBottom = cellTop + tH;
```

```
            int tileIndex; // = 0;
            for (int row = startRow; row <= endRow; row++, cellTop += tH, cellBottom += tH) {
                int cellLeft = startCol * tW + tLx1;
                int cellRight = cellLeft + tW;
                for (int col = startCol; col <= endCol; col++, cellLeft += tW, cellRight += tW) {
                    tileIndex = t.getCell(col, row);
                    if (tileIndex != 0) {
                        int intersectLeft = (sx1 < cellLeft) ? cellLeft : sx1;
                        int intersectTop = (sy1 < cellTop) ? cellTop : sy1;
                        int intersectRight = (sx2 < cellRight) ? sx2
                                : cellRight;
                        int intersectBottom = (sy2 < cellBottom) ? sy2
                                : cellBottom;
                        if (intersectLeft > intersectRight) {
                            int temp = intersectRight;
                            intersectRight = intersectLeft;
                            intersectLeft = temp;
                        }
                        if (intersectTop > intersectBottom) {
                            int temp = intersectBottom;
                            intersectBottom = intersectTop;
                            intersectTop = temp;
                        }
                        int intersectWidth = intersectRight - intersectLeft;
                        int intersectHeight = intersectBottom - intersectTop;
                        int image1XOffset = getImageTopLeftX(intersectLeft,
                                intersectTop, intersectRight, intersectBottom);
                        int image1YOffset = getImageTopLeftY(intersectLeft,
                                intersectTop, intersectRight, intersectBottom);
                        int image2XOffset = t.tileSetX[tileIndex]
                                + (intersectLeft - cellLeft);
                        int image2YOffset = t.tileSetY[tileIndex]
                                + (intersectTop - cellTop);
                        if (doPixelCollision(image1XOffset, image1YOffset,
                                image2XOffset, image2YOffset, this.sourceImage,
                                this.t_currentTransformation, t.sourceImage,
                                TRANS_NONE, intersectWidth, intersectHeight)) {
                            return true;
                        }
                    }
                }
            }
            return false;
        }
    }
```

使用函数 collidesWith(Sprite s, boolean pixelLevel)实现精灵和精灵的碰撞，具体代码如下所示。

```
    public final boolean collidesWith(Sprite s, boolean pixelLevel) {
        if (!(s.visible && this.visible)) {
            return false;
        }
        int otherLeft = s.x + s.t_collisionRectX;
```

```
int otherTop = s.y + s.t_collisionRectY;
int otherRight = otherLeft + s.t_collisionRectWidth;
int otherBottom = otherTop + s.t_collisionRectHeight;
int left = this.x + this.t_collisionRectX;
int top = this.y + this.t_collisionRectY;
int right = left + this.t_collisionRectWidth;
int bottom = top + this.t_collisionRectHeight;
if (intersectRect(otherLeft, otherTop, otherRight, otherBottom, left,
        top, right, bottom)) {
    if (pixelLevel) {
        if (this.t_collisionRectX < 0) {
            left = this.x;
        }
        if (this.t_collisionRectY < 0) {
            top = this.y;
        }
    if ((this.t_collisionRectX + this.t_collisionRectWidth) > this.width) {
            right = this.x + this.width;
        }
    if ((this.t_collisionRectY + this.t_collisionRectHeight) > this.height) {
            bottom = this.y + this.height;
        }
        if (s.t_collisionRectX < 0) {
            otherLeft = s.x;
        }
        if (s.t_collisionRectY < 0) {
            otherTop = s.y;
        }
        if ((s.t_collisionRectX + s.t_collisionRectWidth) > s.width) {
            otherRight = s.x + s.width;
        }
        if ((s.t_collisionRectY + s.t_collisionRectHeight) > s.height) {
            otherBottom = s.y + s.height;
        }
        if (!intersectRect(otherLeft, otherTop, otherRight,
                otherBottom, left, top, right, bottom)) {
            return false;
        }
        int intersectLeft = (left < otherLeft) ? otherLeft : left;
        int intersectTop = (top < otherTop) ? otherTop : top;
        int intersectRight = (right < otherRight) ? right : otherRight;
        int intersectBottom = (bottom < otherBottom) ? bottom: otherBottom;
        int intersectWidth = Math.abs(intersectRight - intersectLeft);
        int intersectHeight = Math.abs(intersectBottom - intersectTop);
        int thisImageXOffset = getImageTopLeftX(intersectLeft,
                intersectTop, intersectRight, intersectBottom);
        int thisImageYOffset = getImageTopLeftY(intersectLeft,
                intersectTop, intersectRight, intersectBottom);
        int otherImageXOffset = s.getImageTopLeftX(intersectLeft,
                intersectTop, intersectRight, intersectBottom);
        int otherImageYOffset = s.getImageTopLeftY(intersectLeft,
```

```
                        intersectTop, intersectRight, intersectBottom);
                    return doPixelCollision(thisImageXOffset, thisImageYOffset,
                        otherImageXOffset, otherImageYOffset, this.sourceImage,
                        this.t_currentTransformation, s.sourceImage,
                        s.t_currentTransformation, intersectWidth,
                        intersectHeight);
                } else {
                return true;
                }
            }
            return false;
        }
```

使用函数 collidesWith(Bitmap image, int x, int y,boolean pixelLevel)用于实现精灵和图片的碰撞，具体代码如下所示。

```
        public final boolean collidesWith(Bitmap image, int x, int y,
                boolean pixelLevel) {
            if (!(this.visible)) {
                return false;
            }
            int otherLeft = x;
            int otherTop = y;
            int otherRight = x + image.getWidth();
            int otherBottom = y + image.getHeight();
            int left = this.x + this.t_collisionRectX;
            int top = this.y + this.t_collisionRectY;
            int right = left + this.t_collisionRectWidth;
            int bottom = top + this.t_collisionRectHeight;
            if (intersectRect(otherLeft, otherTop, otherRight, otherBottom, left,
                    top, right, bottom)) {
                if (pixelLevel) {
                    if (this.t_collisionRectX < 0) {
                        left = this.x;
                    }
                    if (this.t_collisionRectY < 0) {
                        top = this.y;
                    }
                if ((this.t_collisionRectX + this.t_collisionRectWidth) > this.width) {
                        right = this.x + this.width;
                    }
                if ((this.t_collisionRectY + this.t_collisionRectHeight) > this.height) {
                        bottom = this.y + this.height;
                    }
                    if (!intersectRect(otherLeft, otherTop, otherRight,
                            otherBottom, left, top, right, bottom)) {
                        return false;
                    }
                    int intersectLeft = (left < otherLeft) ? otherLeft : left;
                    int intersectTop = (top < otherTop) ? otherTop : top;
                    int intersectRight = (right < otherRight) ? right : otherRight;
                    int intersectBottom = (bottom < otherBottom) ? bottom: otherBottom;
                    int intersectWidth = Math.abs(intersectRight - intersectLeft);
```

```
                int intersectHeight = Math.abs(intersectBottom - intersectTop);
                int thisImageXOffset = getImageTopLeftX(intersectLeft,
                        intersectTop, intersectRight, intersectBottom);
                int thisImageYOffset = getImageTopLeftY(intersectLeft,
                        intersectTop, intersectRight, intersectBottom);
                int otherImageXOffset = intersectLeft - x;
                int otherImageYOffset = intersectTop - y;
                return doPixelCollision(thisImageXOffset, thisImageYOffset,
                        otherImageXOffset, otherImageYOffset, this.sourceImage,
                        this.t_currentTransformation, image, Sprite.TRANS_NONE,
                        intersectWidth, intersectHeight);
            } else {
                return true;
            }
        }
        return false;
    }
```

在上述 3 个函数中，参数 pixelLevel 表示使用像素检测还是矩形检测。矩形检测只需要将精灵对应成相应的矩形范围进行检查，这种检测速度很快，但不是很准确，对于碰撞要求不高的游戏可以使用。同时还可以将一个 Sprite 分解成很多矩形来使用矩形检测以提高准确性。而像素检测则比较准确，但是速度必然会减慢。

④ 实现简单的主角对话。本实例中设计的魔塔游戏主角可以和 NPC 对话来获得一些信息，然后进行游戏。此处准备通过一个浮动的对话框来显示对话内容，这个对话框只是一个矩形框，然后在右边绘制出对应的 NPC 的头像即可。这里的对话内容可以通过前面介绍的 TextUtil 类来实现自动换行。

⑤ 实现精灵旋转和镜像。在游戏开发时，一般都需要使用旋转和镜像。例如，制作一个飞机游戏时，飞机会有几个方向的图片，这样会增加游戏开发出来的包的大小，所以可以制作一个方向的图片，然后通过精灵的旋转方法来将图片在各个方向进行旋转。这里只实现了 90° 的倍数旋转，分别是在 Sprite 类中定义的常量：TRANS_NONE、TRANS_ROT90、TRANS_ROT180、TRANS_ROT270、TRANS_MIRROR、TRANS_MIRROR_ROT90、TRANS_MIRROR_ROT180 和 TRANS_MIRROR_ROT270。可以通过 setTransform()方法来传输这些常量，设置 Sprite 的旋转和镜像。当然，可以查看该方法的具体实现来更深入地理解 Sprite 的旋转和镜像。

⑥ 实现战斗界面。当主角和怪物碰撞时就会发生战斗，这时需要一个界面来显示战斗效果。本实例中的战斗界面很简单，只分别显示玩家和怪物的头像以及属性，包括生命、攻击和防御。战斗界面功能是由文件 FightScreen.java 实现的，其中绘制战斗界面功能的实现代码如下所示。

```
    protected void onDraw(Canvas canvas)
    {
        mcanvas = canvas;
        int tx, ty, tw, th;
        tw = yarin.SCREENW;
        th = yarin.MessageBoxH;
        tx = 0;
        ty = (yarin.SCREENH - yarin.MessageBoxH) / 2;
        showMessage();
        if (!isFighting)
        {
            tu.DrawText(mcanvas);
```

```
            }
            else
            {
                  yarin.drawImage(canvas, orgeImage, 0, ty + (th - GameMap.TILE_WIDTH) / 2, GameMap.TILE_
WIDTH, GameMap.TILE_WIDTH, orgeSrcX, orgeSrcY);
                  yarin.drawImage(canvas, heroImage, (tw - GameMap.TILE_WIDTH), ty + (th - GameMap.TILE_
WIDTH) / 2, GameMap.TILE_WIDTH, GameMap.TILE_WIDTH, 0, 0);
                  paint.setColor(Color.WHITE);
                  //怪物
                  {
                        tx = 40;
                        ty = (yarin.SCREENH - yarin.MessageBoxH) / 2 + 5;
                        yarin.drawString(canvas, "生命:" + orgeHp, tx, ty, paint);
                        yarin.drawString(canvas, "攻击:" + orgeAttack, tx, ty + yarin.TextSize, paint);
                        yarin.drawString(canvas, "防御:" + orgeDefend, tx, ty + 2 * yarin.TextSize, paint);
                  }
                  //主角
                  {
                        String string = "";
                        ty = (yarin.SCREENH - yarin.MessageBoxH) / 2 + 5;
                        string = hero.getHp() + ":生命";
                        yarin.drawString(canvas, string, (tw - 40 - paint.measureText(string)), ty, paint);
                        string = hero.getAttack() + ":攻击";
                        yarin.drawString(canvas, string, (tw - 40 - paint.measureText(string)), ty + yarin.TextSize, paint);
                        string = hero.getDefend() + ":防御";
                        yarin.drawString(canvas, string, (tw - 40 - paint.measureText(string)), ty + 2 * yarin.TextSize,
paint);
                  }
            }
            tick();
      }
      public void showMessage()
      {
            int x = 0;
            int y = (yarin.SCREENH - yarin.MessageBoxH) / 2;
            int w = yarin.SCREENW;
            int h = yarin.MessageBoxH;
            Paint ptmPaint = new Paint();
            ptmPaint.setARGB(255, 0, 0, 0);
            yarin.fillRect(mcanvas, x, y, w, h, ptmPaint);
            ptmPaint = null;
      }
```

每次战斗过后都会得到经验值，增加经验值功能的实现代码如下所示。

```
      private void tick()
      {
            if (orgeHp <= 0)
            {
                  isFighting = false;
                  tu.InitText("得到" + orgeMoney + "个金币    " + "经验值增加" + orgeExperience, 0, (yarin.SCREENH -
yarin.MessageBoxH) / 2, yarin.SCREENW, yarin.MessageBoxH,
                        0x0, 0xff0000, 255, yarin.TextSize);
            }
```

```
            else if (heroFirst == true)
            {
                  orgeHp -= orgeDamagePerBout;
                  if (orgeHp <= 0)
                  {
                        orgeHp = 0;
                  }
            }
            else
            {
                  hero.cutHp(heroDamagePerBout);
            }
            heroFirst = !heroFirst;
      }
```

在文件 Sprite.java 中编写函数 paint(Canvas canvas)，通过此函数将 Sprite 显示在屏幕上，主要实现代码如下所示。

```
      public final void paint(Canvas canvas) {
                        if (canvas == null) {
                  throw new NullPointerException();
            }
            if (visible) {
      drawImage(canvas,this.x,this.y,sourceImage,frameCoordsX[frameSequence[sequenceIndex]],
                              frameCoordsY[frameSequence[sequenceIndex]],
                              srcFrameWidth,
                              srcFrameHeight);
            }
      }
```

注意： Sprite 的编号是从 0 开始的，但是 TiledLayer 却是从 1 开始的。在 TiledLayer 中，序号 0 表示一个空白的元素（例如，在某个位置什么都不想画，那就把它设置成 0）。Sprite 只由一个单元组成，所以如果想不显示这个单元，设置成 setVisible(false)即可，因而 Sprite 不需要一个特殊的编号来表示空白的单元。

3．游戏音效

音效在游戏开发中起了很重要的作用，在开发游戏时，人们常常忽视游戏的音效。开发者往往把主要精力花费在游戏的图像和动画等方面，而忽视了背景音乐和声音效果。这种做法是不可取的，因为好的游戏音效和音乐可以使玩家融入游戏世界，产生共鸣。音效的作用还不仅限于此。如果没有高超的游戏音效的映衬，再好的图像技巧也无法使游戏的表现摆脱平庸，对玩家也没有足够的吸引力。游戏中的音效可分为如下几类：背景音乐、剧情音乐、音效（动作的音效、使用道具音效、辅助音效）等。背景音乐一般需要一直播放，而剧情音乐则只需要在剧情需要时播放，音效则是很短小的一段，如挥刀的声音、怪物叫声等。本实例中准备为此游戏添加两个背景音乐，一个是菜单背景音乐，另一个是游戏中的背景音乐，操作流程如下所示。

（1）准备两个符合游戏剧情的背景音乐添加到 res\raw 目录下。

（2）创建一个 CMIDIPlayer 类，控制音乐播放。

因为在 Android 中是通过 MediaPlayer 来播放音乐的，所以在类 CMIDIPlayer 中需要构建一个 MediaPlayer 对象，通过 MediaPlayer.create 来装载音乐文件。上述功能的实现文件是 CMIDIPlayer.java，

主要实现代码如下所示。

```
public class CMIDIPlayer
{
	public MediaPlayer	playerMusic;
	public MagicTower	magicTower = null;
	public CMIDIPlayer(MagicTower magicTower)
	{
		this.magicTower = magicTower;
	}
	//播放音乐
	public void PlayMusic(int ID)
	{
		FreeMusic();
		switch (ID)
		{
			case 1:
				//装载音乐
				playerMusic = MediaPlayer.create(magicTower, R.raw.menu);
				//设置循环
				playerMusic.setLooping(true);
				try
				{
					//准备
					playerMusic.prepare();
				}
				catch (IllegalStateException e)
				{
					e.printStackTrace();
				}
				catch (IOException e)
				{
					e.printStackTrace();
				}
				//开始
				playerMusic.start();
				break;
			case 2:
				playerMusic = MediaPlayer.create(magicTower, R.raw.run);
				playerMusic.setLooping(true);
				try
				{
					playerMusic.prepare();
				}
				catch (IllegalStateException e)
				{
					e.printStackTrace();
				}
				catch (IOException e)
				{
					e.printStackTrace();
				}
				playerMusic.start();
				break;
		}
```

```
        }
        //退出释放资源
        public void FreeMusic()
        {
                if (playerMusic != null)
                {
                        playerMusic.stop();
                        playerMusic.release();
                }
        }
        //停止播放
        public void StopMusic()
        {
                if (playerMusic != null)
                {
                        playerMusic.stop();
                }
        }
}
```

在上述代码中，通过 PlayMusic 加上 ID 来确定播放什么音乐，通过 StopMusic 来停止正在播放的音乐，当程序退出时调用 FreeMusic()方法来释放播放音乐产生的资源。

（3）创建“是否开启音效”界面。

在游戏中不可能强制玩家接受要播放的音乐，所以设计一个界面供玩家选择是否开启音乐很有必要，播放音乐代码如下所示。

```
mCMIDIPlayer.PlayMusic(1);
```

mCMIDIPlayer 是本实例中构建的一个 CMIDIPlayer 类的对象。在进入游戏时，需要播放另一首背景音乐，和上面的代码一样，只需要通过参数的 ID 来设置要播放的音乐。

（4）释放资源。当退出游戏时，需要调用类 CMIDIPlayer 的方法 FreeMusic()来释放资源，主要代码如下所示。

```
mCMIDIPlayer.FreeMusic();
```

在大多数游戏中，控制音效的界面也不止一个，可以在主菜单界面中设置音效，同样还可以在游戏中通过一个弹出菜单等来控制音效，但是实现方式都相同，读者可以自己将其完善，尽可能地方便用户随时控制音乐开关。

执行后的界面效果如图 11-3 所示。

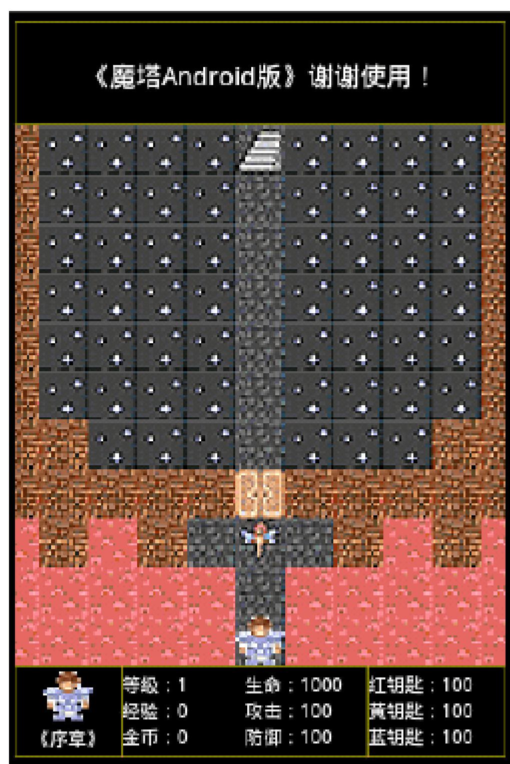


图 11-3　执行效果

11.3　纸牌类游戏

实例 130	纸牌类游戏
源码路径	光盘:\daima\130
视频路径	光盘:\视频\130
实例必备	130.游戏中的数学.pdf ① 坐标系 ② 矢量（向量）

11.3.1　实例说明

在本游戏中，预先准备了 3 张扑克牌，分别是 A、2、3，如图 11-4 所示。



图 11-4　3 张扑克牌

本游戏的玩法是：用户从这 3 张扑克牌中找出 A，无论正确还是错误都会输出对应的提示。

11.3.2　具体实现

本实例的实现文件是 puke.java，具体实现流程如下所示。

（1）定义 Activity 类的子类 puke，首先声明需要的控件对象，然后分别建立循环消息队列和获取控件实例，主要代码如下所示。

```
public class puke extends Activity {
    /** Called when the activity is first created.*/
    //声明控件
     private TextView outPut;
     private ImageView imageFace;
     private ImageView image_View1;
     private ImageView image_View2;
     private ImageView image_View3;
     private ProgressBar progressBar;
     private Button button_start;
     private Button button_end;
     private MyHandler myHandler;          //进度条线程
     private int i;                        //进度条参数

    private static int[ ] poker = {R.drawable.p_1,R.drawable.p_2,R.drawable.p_3};
```

```
 @Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //建立循环消息队列
    HandlerThread handlerThread = new HandlerThread("thread");
    //很重要，必须要写
    handlerThread.start();
    myHandler = new MyHandler(handlerThread.getLooper());
    //取得控件实例
    outPut = (TextView)findViewById(R.id.outPut);
    imageFace = (ImageView)findViewById(R.id.imageface);
    image_View1 = (ImageView)findViewById(R.id.image1);
    image_View2 = (ImageView)findViewById(R.id.image2);
    image_View3 = (ImageView)findViewById(R.id.image3);
    progressBar = (ProgressBar)findViewById(R.id.progressbar);
    button_start = (Button)findViewById(R.id.start);
    button_end = (Button)findViewById(R.id.end);
    //洗牌
    random();
```

（2）编写选择第一张牌后的响应事件，选到 A 则输出“WOW，选对了哦，真厉害！你是怎么做到的？”，没选到则输出“真遗憾~~，这次运气不好，再来一次吧？”。具体代码如下所示。

```
    //牌 1 监听器
    image_View1.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                image_View1.setImageDrawable(getResources().getDrawable(poker[0]));
                image_View2.setImageDrawable(getResources().getDrawable(poker[1]));
                image_View3.setImageDrawable(getResources().getDrawable(poker[2]));
                image_View2.setAlpha(100);    //设置没被选中的牌渐隐效果
                image_View3.setAlpha(100);

                if(poker[0] == R.drawable.p_1){

                    outPut.setText("WOW，选对了哦，真厉害！你是怎么做到的？");
 imageFace.setImageDrawable(getResources().getDrawable(R.drawable.qq_suprise));

                }else{

                    outPut.setText("真遗憾~~，这次运气不好，再来一次吧？");
 imageFace.setImageDrawable(getResources().getDrawable(R.drawable.qq_despise));

                }
            }

    });
```

（3）编写选择第二张牌后的响应事件，选到 A 则输出“WOW，选对了哦，真厉害！你是怎么做

到的？”，没选到则输出“真遗憾~~，这次运气不好，再来一次吧？”。具体代码如下所示。

```
        image_View2.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

                image_View1.setImageDrawable(getResources().getDrawable(poker[0]));
                image_View2.setImageDrawable(getResources().getDrawable(poker[1]));
                image_View3.setImageDrawable(getResources().getDrawable(poker[2]));

                image_View1.setAlpha(100);
                image_View3.setAlpha(100);

                if(poker[1] == R.drawable.p_1){

                    outPut.setText("WOW，选对了哦，真厉害！你是怎么做到的？");

    imageFace.setImageDrawable(getResources().getDrawable(R.drawable.qq_suprise));
                } else {
                    outPut.setText("真遗憾~~，这次运气不好，再来一次吧？");
                }
        });
```

（4）编写选择第三张牌后的响应事件，选到 A 则输出“WOW，选对了哦，真厉害！你是怎么做到的？”，没选到则输出“真遗憾~~，这次运气不好，再来一次吧？”。具体代码如下所示。

```
        image_View3.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

                image_View1.setImageDrawable(getResources().getDrawable(poker[0]));
                image_View2.setImageDrawable(getResources().getDrawable(poker[1]));
                image_View3.setImageDrawable(getResources().getDrawable(poker[2]));

                image_View1.setAlpha(100);
                image_View2.setAlpha(100);

                if(poker[2] == R.drawable.p_1){

                    outPut.setText("WOW，选对了哦，真厉害！你是怎么做到的？");

    imageFace.setImageDrawable(getResources().getDrawable(R.drawable.qq_suprise));

                }else{

                    outPut.setText("真遗憾~~，这次运气不好，再来一次吧？");

    imageFace.setImageDrawable(getResources().getDrawable(R.drawable.qq_despise));

                }
```

```
            }

        });
```

（5）编写单击“洗牌”按钮事件的处理方法 onClick(View v)，设置进度条可见，并启动进度条线程，具体代码如下所示。

```
            public void onClick(View v) {
                // TODO Auto-generated method stub
                outPut.setText("猜猜看黑桃 A 是哪一张？");
                //设置进度条可见，启动进度条线程
                progressBar.setVisibility(View.VISIBLE);
                myHandler.post(progressBarThread);

                imageFace.setImageDrawable(getResources().getDrawable(R.drawable.qq_laugh));
                image_View1.setImageDrawable(getResources().getDrawable(R.drawable.poker_back));
                image_View2.setImageDrawable(getResources().getDrawable(R.drawable.poker_back));
                image_View3.setImageDrawable(getResources().getDrawable(R.drawable.poker_back));

                image_View1.setAlpha(255);
                image_View2.setAlpha(255);
                image_View3.setAlpha(255);

                random();
            }

        });
```

（6）编写单击“退出”按钮事件的处理方法，具体代码如下所示。

```
        //end 键结束程序
        button_end.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                finish();
            }

        });
```

（7）编写随机洗牌处理方法 random()，具体代码如下所示。

```
    //随机洗牌方法
    private void random(){
        Random rand = new Random();
        int temp1 = 0;
        int temp2 = 0;
        for(int i=0; i<3; i++){
            temp1 = rand.nextInt(3);
            temp2 = poker[i];
            poker[i] = poker[temp1];
```

```
                poker[temp1] = temp2;
            }

    }
```

（8）编写进度条线程代码，具体代码如下所示。

```
    //进度条线程
    Runnable progressBarThread = new Runnable(){
        @Override
        public void run() {
            // TODO Auto-generated method stub
            i+=10;

            try{
                Thread.sleep(20);
            }catch(InterruptedException e){
                e.printStackTrace();
            }

            Message msg = myHandler.obtainMessage();
            msg.arg1 = i;
            myHandler.sendMessage(msg);
        }

    };
    //设置进度条不可见
    Runnable progressInvisible = new Runnable(){
        @Override
        public void run() {
            // TODO Auto-generated method stub
            progressBar.setVisibility(View.INVISIBLE);
        }

    };
```

至此，本实例的主要功能介绍完毕。执行后的初始界面效果如图 11-5 所示，选对了的界面效果如图 11-6 所示。

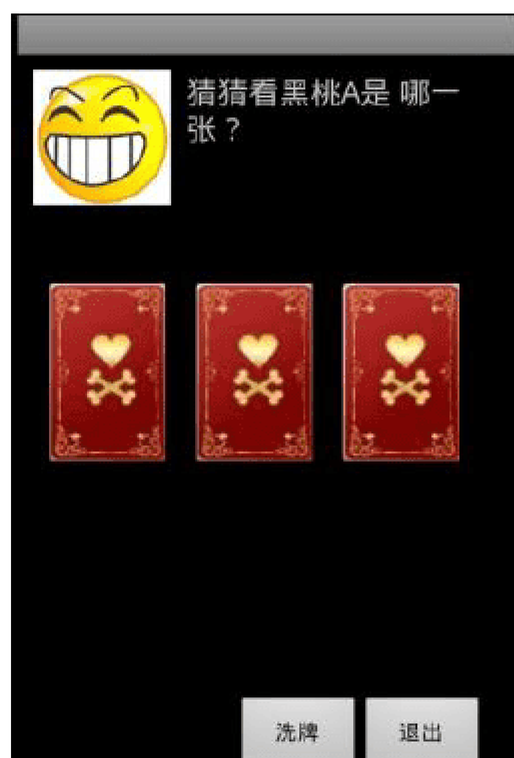


图 11-5　初始效果



图 11-6　选对界面效果

11.4　体育竞技类游戏——疯狂足球

实例 131	开发一个足球游戏
源码路径	光盘:\daima\131
视频路径	光盘:\视频\131
实例必备	131.游戏中的物理.pdf ① 速度 ② 加速度 ③ 位移 ④ 重力系统

11.4.1　实例说明

足球游戏是指以足球作为游戏主题的游戏，目前在市场上主要分为足球类小游戏、足球类网页游戏、足球类电视游戏、足球类单机游戏等几大类别。在足球游戏中，用户可以扮演不同的角色，扮演球员角色的通常以操作类的足球游戏为主，代表作包括 FIFA 系列、实况足球系列。用户也可以扮演经理人角色，代表作分别是足球经理人系列游戏。不同类型的足球游戏，可以让玩家得到不同的体验。足球游戏平台包括 PSP、PS3、PS2、NDSL、PC、手机、XBOX360 等，游戏视图包括屏幕视图、道具视图、角色视图等。

11.4.2　具体实现

在 Android 中，Activity 类负责切换不同界面。在本实例中，Activity 除了负责切换不同界面外，还能够实现按键单击和修改按键状态的功能。本实例的 Activity 类是由文件 FootballActivity. java 实现的，其主要实现代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {                    //重写 onCreate()方法
    super.onCreate(savedInstanceState);
    initWelcomeSound(this);                                          //初始化声音库
    requestWindowFeature(Window.FEATURE_NO_TITLE);                   //设置全屏
    getWindow().setFlags(
          WindowManager.LayoutParams.FLAG_FULLSCREEN,
          WindowManager.LayoutParams.FLAG_FULLSCREEN
          );
    welcome = new WelcomeView(this);                                 //将屏幕切换到欢迎界面
    setContentView(welcome);
    current = welcome;
    if(wantSound && mpWelcomeMusic!=null){                           //如需要，播放相应声音
      mpWelcomeMusic.start();
    }
    initRects();                                                     //初始化用于匹配单击事件的矩形框
}
```

```
    //初始化欢迎界面的声音
    public void initWelcomeSound(Context context){
        mpWelcomeMusic = MediaPlayer.create(context, R.raw.music);
    }
    //初始化矩形框
    public void initRects(){
     rectPlus = new Rect[3];
     rectMinus = new Rect[3];
     for(int i=0;i<3;i++){
        rectPlus[i] = new Rect(244,200+40*i,280,236+40*i);
        rectMinus[i] = new Rect(280,200+40*i,316,236+40*i);
     }
     rectSound = new Rect(135,370,185,420);
     rectStart = new Rect(205,425,295,475);
     rectQuit = new Rect(25,425,115,475);
     rectGallery = new Rect(10,10,310,110);
    }
     //重写 onTouchEvent()方法
     public boolean onTouchEvent(MotionEvent event) {
        if(event.getAction()== MotionEvent.ACTION_UP){                //判断事件类型
            int x = (int)event.getX();                                //获得单击处的 x 坐标
            int y = (int)event.getY();                                //获得单击处的 y 坐标
            if(current == welcome){                                   //如果当前界面是欢迎界面
                if(rectGallery.contains(x, y)){                       //用户单击的是 Gallery
                    welcome.cg.galleryTouchEvnet(x, y);               //交给 Gallery 来处理单击事件
                }
                else if(rectSound.contains(x, y)){                    //按下的是声音选项
                    this.wantSound = !this.wantSound;                 //更改声音选项
                    return true;
                }
                else if(rectStart.contains(x, y)){                    //按下开始键
                    if(checkLayout(welcome.layout)){                  //检查玩家选择的布局是否正确
                        layoutArray = welcome.layout;                 //获得玩家选择站位布局
                        lv = new LoadingView(this);         //创建读取进度 View
                        this.setContentView(lv);            //将屏幕设为读取进度的 LoadingView
                        this.current = lv;                  //记录当前 View
                        lv.lt.start();                      //启动 LoadingView 的刷屏线程
                        new Thread(){                       //启动一个新线程，在其中创建 GameView 对象
                            public void run(){
                                Looper.prepare();
                                if(wantSound){
                                    initSound();            //初始化声音
                                }
                                //创建游戏界面
gv = new GameView(FootballActivity.this,imageIDs[welcome.cg.currIndex]);
                                lv.progress = 100;
                                welcome = null;             //释放 WelcomeView
                            }
                        }.start();
                    }
                }
```

```
            else if(rectQuit.contains(x,y)){               //按下退出键
                System.exit(0);                            //程序退出
            }
            else{                   //检查是否按下了修改队员站位的加号和减号按钮
        for(int i=0;i<3;i++){
        if(rectPlus[i].contains(x,y)){      //如果有加号按钮点下，就增加对应进攻防守线上人数
                                            //如果有剩余的人再加
                if(welcome.layout[0]+welcome.layout[1]+welcome.layout[2] <10){
                        welcome.layout[i]++;
                    }
                    break;
                }
                if(rectMinus[i].contains(x, y)){          //如果有减号按钮点下，就减少相应人数
                    if(welcome.layout[i] > 0){  //如果该处人数不为 0，就减少一个
                        welcome.layout[i]--;
                    }
                    break;
                }
            }
        }
    }
    else if(current == gv){                              //如果当前显示的 View 为 GameView
        if(gv.rectMenu.contains(x,y)){                   //如果按下了菜单按钮
            gv.isShowDialog = true;                      //设置显示对话框
            gv.ball.isPlaying = false;                   //足球停止移动
            pmt.flag = false;                            //使 PlayerMoveThread 空转
        }
    else if(gv.rectYesToDialog.contains(x,y)){           //如果按下的是对话框中的“是”按钮
    if(gv.isShowDialog){                                 //检查对话框是不是正在显示
            welcome = new WelcomeView(this);             //新建一个 WelcomeView
                setContentView(welcome);                 //设置当前屏幕为 WelcomeView
                welcome.status = 3;                      //直接设为待命状态
                current = welcome;                       //记录当前屏幕
                gv = null;                               //将 GameView 指向的对象声明为垃圾
                if(wantSound && mpWelcomeMusic!=null){
                  //如需要，播放声音
                  mpWelcomeMusic.start();
                }
            }
        }
else if(gv.rectNoToDialog.contains(x,y)){                //如果按下的是对话框中的“否”按钮
            if(gv.isShowDialog){                         //检查对话框是不是正在显示
                gv.isShowDialog = false;                 //不显示对话框
                pmt.flag = true;                         //设置双方球员可移动
                gv.ball.isPlaying = true;                //设置足球可移动
            }
        }
    }
    else if(current == lv){                              //如果当前屏幕为 LoadingView
        if(lv.progress == 100){                          //如果进度达到 100%
            setContentView(gv);                          //屏幕切换到 GameView
```

```
                        current = gv;                    //记录当前 View
                        lv = null;                       //lv 指向的对象声明为垃圾
                        if(mpWelcomeMusic.isPlaying()){  //如需要，播放相应声音
                            mpWelcomeMusic.stop();
                        }
                        gv.startGame();                  //开始游戏
                    }
                }
            }
            return true;
        }
        //加载游戏中用到的声音
        public void initSound(){
            mpKick = MediaPlayer.create(this, R.raw.kick);
            updateProgressView();                        //更新进度条
            mpCheerForWin = MediaPlayer.create(this, R.raw.cheer_win);
            updateProgressView();                        //更新进度条
            mpCheerForLose = MediaPlayer.create(this, R.raw.cheer_lose);
            updateProgressView();                        //更新进度条
            mpCheerForGoal = MediaPlayer.create(this, R.raw.cheer_goal);
            updateProgressView();                        //更新进度条
            mpLargerGoal = MediaPlayer.create(this, R.raw.lager_goal);
            updateProgressView();                        //更新进度条
            mpIce = MediaPlayer.create(this, R.raw.ice);
            updateProgressView();                        //更新进度条
        }
        //更新进度条的进度
        public void updateProgressView(){
         lv.progress+=15;
        }
        @Override
        //检查用户输入的 layout 是否合法
        public boolean checkLayout(int[ ] layout){
            int sum=0;
            for(int i=0;i<layout.length;i++){            //遍历存放球员站位的数组
                if(layout[i]<0){                         //如果发现某个进攻/防守阵线上的球员为负数
                    return false;
                }
                else{
                    sum+=layout[i];                      //将各个阵线上的球员个数相加
                }
            }
            if(sum == 10){                               //如果和为 10，则该站位合法
                return true;
            }
            else{
                return false;                            //返回 false
            }
        }
}
```

执行代码后的初始界面如图 11-7 所示，游戏界面如图 11-8 所示。

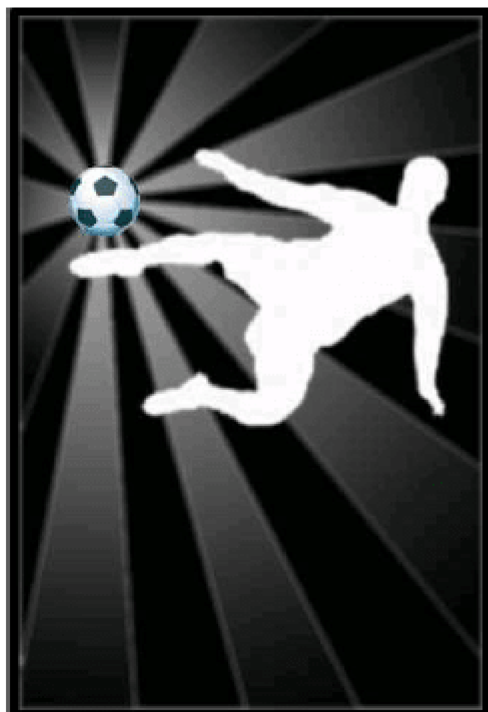


图 11-7　初始界面

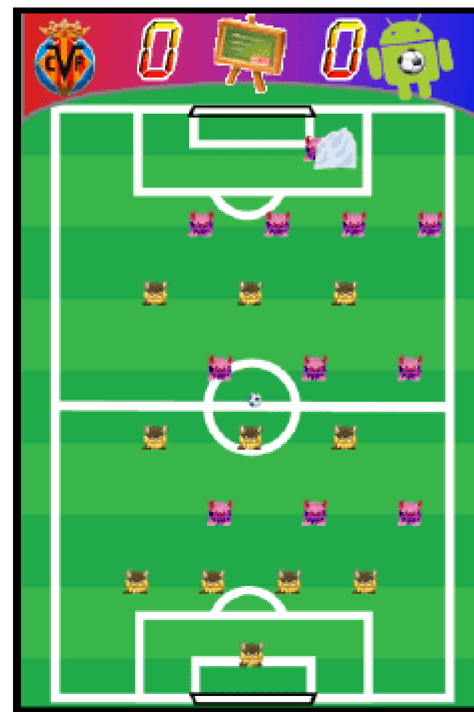


图 11-8　游戏界面

第 12 章　移动 Web 应用

从移动电话的产生，到当前移动互联应用的风生水起，我们步入到任何人都有机会获得大量信息资源的移动互联网时代。尽管移动计算技术已扮演了如此重要的角色，但仍处于发展初期。对于需要吸引不同群体用户、满足不同业务需求的应用而言，如何使用一个实用、价格合理，且可支持大量应用的方式来实现我们的移动愿景？在很多情况下看来，答案是使用 Web 技术。本章将通过几个典型实例的实现过程，详细介绍为 Android 系统开发移动 Web 应用程序的过程。

12.1　编写第一个网页

实例 132	为 Android 编写第一个网页
源码路径	光盘:\daima\132
视频路径	光盘:\视频\132
实例必备	132.Web 开发标准介绍.docx.pdf ① Web 开发标准概述 ② 为什么要使用 Web 标准

12.1.1　实例说明

下面以一个具体例子作为开始，详细讲解在 Android 平台中使用 HTML+CSS+JavaScript 设计一个网页的基本知识，并讲解在 Android 设备中调试运行的具体方法。本实例假设有一个很好的网页，广大用户已经浏览了很多次。

12.1.2　具体实现

主页文件 index.html 的源代码如下所示。

```
<html>
    <head>
        <title>aaa</title>
        <link rel="stylesheet" href="desktop.css" type="text/css" />
    <body>
        <div id="container">
          <div id="header">
                <h1><a href="./">好东西要分享</a></h1>
                <div id="utility">
                    <ul>
                        <li><a href="about.html">关于我们</a></li>
```

```
                    <li><a href="blog.html">博客</a></li>
                    <li><a href="contact.html">联系我们</a></li>
                </ul>
            </div>
            <div id="nav">
                <ul>
                    <li><a href="bbb.html">发邮件吧</a></li>
                    <li><a href="ccc.html">电话支持</a></li>
                    <li><a href="ddd.html">在线客服</a></li>
                    <li><a href="http://www.aaa.com">在线视频</a></li>
                </ul>
            </div>
        </div>
        <div id="content">
            <h2>关于我们</h2>
            <p>这是一个学习的网站，也是一个交流的网站.....</p>
        </div>
        <div id="sidebar">
            <img alt="好图片" src="aaa.png">
            <p>这是一个学习的网站，也是一个交流的网站.....</p>
      </div>
        <div id="footer">
            <ul>
                <li><a href="bbb.html">服务</a></li>
                <li><a href="ccc.html">关于我们</a></li>
                <li><a href="ddd.html">博客</a></li>
            </ul>
            <p class="subtle">世界第一</p>
        </div>
    </div>
  </body>
</html>
```

根据“样式和表现相分离”的原则，需要单独写一个 CSS 文件，通过这个 CSS 文件来给上述网页进行修饰，修饰的最终目的是能够在 Android 手机上浏览。

注意： 在现实的开发应用中，最好将桌面浏览器的样式表和 Android 样式表划清界限。笔者认为，写两个完全独立的文件会舒服很多。当然，还有另一种做法是把所有的 CSS 规则放到一个单一的样式表中，但是这种做法不值得提倡，原因有两方面。

☑ 文件太长了就显得麻烦，不利于维护。

☑ 把太多不相关的桌面样式规则发送到手机上，会浪费一些宝贵的带宽和存储空间。

开始写 CSS 文件，为了适应 Android 系统，加入下面的 link 标签。

```
<link rel="stylesheet" type="text/css"
 href="android.css" media="only screen and (max-width: 480px)" />
<link rel="stylesheet" type="text/css"
 href="desktop.css" media="screen and (min-width: 481px)" />
```

在上述代码中，最明显的是浏览器宽度的变化，即

```
max-width: 480px
min-width: 481px
```

这是因为手机屏幕的宽度和计算机屏幕的宽度不一样（当然长度也不一样，但是都具有下拉功能），480 是 Android 系统的标准宽度，上述代码的功能是不管浏览器的窗口有多大，桌面用户看到的都是文件 desktop.css 中样式修饰的页面，宽度都是用如下代码设置的。

```
max-width: 480px
min-width: 481px
```

本实例涉及两个 CSS 文件，一个是 desktop.css，此文件是在开发计算机页面时编写的样式文件，为 HTML 页面服务。而文件 Android.css 是一个新文件，通过它可以将上面的网页显示在 Android 手机中。当读者开发出完整的 Android.css 后，可以直接在 HTML 文件中将如下代码删除，即不再用这个修饰文件。

```
<link rel="stylesheet" type="text/css"
 href="desktop.css" media="screen and (min-width: 481px)" />
```

此时在 Chrome 浏览器中浏览修改后的 HTML 文件，不管从 Android 手机浏览器还是计算机浏览器，执行后都将得到一个完整的页面展示，此时的完整代码如下所示。

```
<html>
    <head>
        <title>AAAA</title>
        <link rel="stylesheet" type="text/css" href="android.css" media="only screen and (max-width: 480px)" />
        <link rel="stylesheet" type="text/css" href="desktop.css" media="screen and (min-width: 481px)" />
        <!--[if IE]>
            <link rel="stylesheet" type="text/css" href="explorer.css" media="all" />
        <![endif]-->
        <script type="text/javascript" src="jquery.js"></script>
        <script type="text/javascript" src="android.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    </head>
    <body>
        <div id="container">
          <div id="header">
                <h1><a href="./">好东西要分享</a></h1>
                <div id="utility">
                    <ul>
                        <li><a href="about.html">关于我们</a></li>
                        <li><a href="blog.html">博客</a></li>
                        <li><a href="contact.html">联系我们</a></li>
                    </ul>
                </div>
                <div id="nav">
                    <ul>
                        <li><a href="bbb.html">发邮件吧</a></li>
                        <li><a href="ccc.html">电话支持</a></li>
                        <li><a href="ddd.html">在线客服</a></li>
                        <li><a href="http://www.aaa.com">在线视频</a></li>
                    </ul>
                </div>
            </div>
            <div id="content">
                <h2>关于我们</h2>
                <p>这是一个学习的网站，也是一个交流的网站.....</p>
            </div>
```

```
                <div id="sidebar">
                    <img alt="好图片" src="aaa.png">
                    <p>这是一个学习的网站，也是一个交流的网站.....</p>
              </div>
                <div id="footer">
                    <ul>
                        <li><a href="bbb.html">服务</a></li>
                        <li><a href="ccc.html">关于我们</a></li>
                        <li><a href="ddd.html">博客</a></li>
                    </ul>
                    <p class="subtle">世界第一</p>
                </div>
            </div>
        </body>
</html>
</html>
```

而 desktop.css 的代码如下所示。

```
For example:
body {
    margin:0;
    padding:0;
    font: 75% "Lucida Grande", "Trebuchet MS", Verdana, sans-serif;
}
```

执行效果如图 12-1 所示。



图 12-1　执行效果

为了使页面变得更加精彩，本实例添加了一些充满视觉效果的样式，具体实现流程如下所示。

（1）给 header 文字加 1px 向下的白色阴影，背景加上 CSS 渐变效果，具体代码如下所示。

```
#header h1 a {
    text-shadow: 0px 1px 1px #fff;
    background-image: -webkit-gradient(linear, left top, left bottom, from(#ccc), to(#999));
}
```

对于上述代码有两点说明。

☑ text-shadow：参数从左到右分别表示水平偏移、垂直偏移、模糊效果和颜色。在大多数情况下，可以将文字设置成上面代码中的数值，这在 Android 界面中的显示效果也不错。在大部

分浏览器上，将模糊范围设置为 0px 也能看到效果。但 Andorid 要求模糊范围最少是 1px，如果设置成 0px，则在 Android 设备上将显示不出来文字阴影。

☑ -webkit-gradient：功能是让浏览器在运行时产生一张渐变的图片。因此，可以把 CSS 渐变功能用在任何平常指定图片（如背景图片或者列表式图片）url 的地方。参数从左到右的排列顺序分别是渐变类型（可以是 linear 或者 radial 的）、渐变起点（可以是 left top、left bottom、right top 或者 right bottom）、渐变终点、起点颜色和终点颜色。

注意： 在上述赋值中，不能颠倒描述渐变起点、终点常量（left top、left bottom、right top、right bottom）的水平和垂直顺序。也就是说，top left、bottom left、top right 和 bottom right 是不合法的值。

（2）给导航菜单加上圆角样式，代码如下所示。

```
#header ul li:first-child a {
    -webkit-border-top-left-radius: 8px;
    -webkit-border-top-right-radius: 8px;
}
#header ul li:last-child a {
    -webkit-border-bottom-left-radius: 8px;
    -webkit-border-bottom-right-radius: 8px;
}
```

上述代码使用-webkit-border-radius 属性描述角的方式，定义列表第一个元素的上两个角和最后一个元素的下两个角为以 8px 为半径的圆角。此时在 Android 中的执行效果如图 12-2 所示。



图 12-2　在 Android 中的执行效果

此时会发现列表显示样式变为了圆角样式，整个外观显得更加圆滑和自然。

12.2　使用 jQuery 设计网页

实例 133	在 Android 中使用 jQuery 设计网页
源码路径	光盘:\daima\133
视频路径	光盘:\视频\133
实例必备	133.jQuery 介绍.pdf

12.2.1 实例说明

jQuery 是继 prototype 之后又一个优秀的 JavaScript 框架。它是轻量级的 js 库，兼容 CSS 3，还兼容各种浏览器（IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+），jQuery 2.0 及后续版本不再支持 IE6/7/8 浏览器。jQuery 使用户能更方便地处理 HTML documents、events，实现动画效果，并且方便地为网站提供 Ajax 交互。jQuery 还有一个比较大的优势，其文档说明很全，而且各种应用也很详细，同时还有许多成熟的插件可供选择。jQuery 能够使用户的 HTML 页面保持代码和 HTML 内容分离，也就是说，不用再在 HTML 中插入大量 JS 脚本来调用命令了，只需定义 ID 即可。

12.2.2 具体实现

（1）隐藏 header 中的 ul 元素，使其在用户第一次浏览页面之后不会显示出来，具体代码如下所示。

```
#header ul．hide{
display:none;
}
```

（2）定义显示和隐藏菜单的按钮，代码如下所示。

```
<div class="leftButton"onclick="toggleMenu()">Menu</div>
```

定义一个带有 leftButton 类的 div 元素，将其放在 header 中，下面是这个按钮的完整 CSS 样式代码。

```
#header div.leftButton {
    position: absolute;
    top: 7px;
    left: 6px;
    height: 30px;
    font-weight: bold;
    text-align: center;
    color: white;
    text-shadow: rgba (0,0,0,0.6) 0px -1px 1px;
    line-height: 28px;
    border-width: 0 8px 0 8px;
    -webkit-border-image: url(images/button.png) 0 8 0 8;
}
```

上述代码的具体说明如下。

- ☑ position: absolute：从顶部开始，设置 position 为 absolute，相当于把这个 div 元素从 HTML 文件流中去掉，从而可以设置最上面和最左面的坐标。
- ☑ height: 30px：设置高度为 30px。
- ☑ font-weight: bold：定义文字格式为粗体，白色带有一点向下的阴影，在元素里居中显示。
- ☑ text-shadow: rgba：rgb（255，255，255）、rgb（100%，100%，100%）格式和#FFFFFF 格式是一个原理，都是设置颜色值的。在 rgba()函数中，其第 4 个参数用来定义 alpha 值（透明度），取值范围为 0～1。其中，0 表示完全透明，1 表示完全不透明，0～1 之间的小数表示不同程度的半透明。
- ☑ line-height：元素中的文字往下移动的距离，使之不会和上边框齐平。
- ☑ border-width 和-webkit-border-image：这两个属性一起决定把一张图片的一部分放入某一元素

的边框中。如果元素大小由于文字的增减而改变，图片会自动拉伸适应这样的变化。这一点其实非常棒，意味着只需要不多的图片、少量的工作、低带宽和更少的加载时间。

☑ border-width：让浏览器把元素的边框定位在距上 0px、距右 8px、距下 0px、距左 8px 的位置（4 个参数从上开始，以顺时针为序）。不需要指定边框的颜色和样式。

☑ url(images/button.png) 0 8 0 8：5 个参数从左到右分别是：图片的 URL、上边距、右边距、下边距、左边距（再一次，从上顺时针开始）。URL 可以是绝对（如 http://example.com/myBorderImage.png）或者相对路径，后者相对于样式表所在的位置，而不是引用样式表的 HTML 页面的位置。

（3）开始在 HTML 文件中插入引入 JavaScript 的代码，将对 aaa.js 和 bbb.js 的引用写到 HTML 文件中。

```
<script type="text/javascript" src="aaa.js"></script>
<script type="text/javascript" src="bbb.js"></script>
```

在文件 bbb.js 中编写一段 JavaScript 代码，这段代码的主要作用是让用户显示或者隐藏 nav 菜单。代码如下所示。

```
if (window.innerWidth && window.innerWidth <= 480) {
    $(document).ready(function(){
        $('#header ul').addClass('hide');
        $('#header').append('<div class="leftButton" onclick="toggleMenu()">Menu</div>');
    });
    function toggleMenu() {
        $('#header ul').toggleClass('hide');
        $('#header .leftButton').toggleClass('pressed');
    }
}
```

对上述代码的具体说明如下所示。

☑ 第 1 行：括号中的代码表示当 window 对象的 innerWidth 属性存在并且 innerWidth 小于等于 480px（这是大部分手机合理的最大宽度值）时才执行到内部。这一行保证只有当用户用 Android 手机或者类似大小的设备访问这个页面时，上述代码才会执行。

☑ 第 2 行：使用了函数(document).ready，此函数是网页加载完成函数。这段代码的功能是设置当网页加载完成之后才运行其中的代码。

☑ 第 3 行：使用了典型的 jQuery 代码，目的是选择 header 中的 ul 元素并且向其中添加 hide 类开始。

此处的 hide 作用于前面 CSS 文件中的选择器，这行代码执行的效果是隐藏 header 中的 ul 元素。

☑ 第 4 行：此处是给 header 添加按钮的地方，目的是显示和隐藏菜单。

☑ 第 7 行：函数 toggleMenu()用 jQuery 的 toggleClass()函数来添加或删除所选择对象中的某个类。这里应用了 header 中 ul 里的 hide 类。

☑ 第 8 行：在 header 的 leftButton 里添加或删除 pressed 类，类 pressed 的具体代码如下所示。

```
#header div.pressed {
    -webkit-border-image: url(images/button_clicked.png) 0 8 0 8;
}
```

通过上述样式和 JavaScript 行为设置以后，Menu 开始动起来了，默认是隐藏了链接内容，单击之后才会在下方显示链接信息，如图 12-3 所示。



图 12-3　下方显示信息

12.3　使用页面模板

实例 134	在 Android 中使用页面模板
源码路径	光盘:\daima\134
视频路径	光盘:\视频\134
实例必备	134.组件的增强样式.pdf

12.3.1　实例说明

在日常生活中已经离不开天气预报，无论是远行旅游还是上班，都根据天气预报决定是否需要带伞和增减衣物。本实例是一个天气预报程序，能够根据所选择的城市显示天气情况。和前面实例的区别是要在平板上浏览程序，所以分辨率的大小较之前有所差别。

12.3.2　具体实现

实例文件 template.html 的具体代码如下所示。

```
<!DOCTYPE html>
<html>
	<head>
	<meta charset="utf-8">
	<title>Page Template</title>
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
	<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
	<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
</head>
<body>
<div data-role="page">
	<div data-role="header">
		<h1>页头</h1>
```

```
        </div>
        <div data-role="content">
            <p>你好 jQuery Mobile!</p>
        </div>
        <div data-role="footer" data-position="fixed">
            <h4>页尾</h4>
        </div>
</div>
</body>
</html>
```

将上述 HTML 文件在台式机中运行后的效果如图 12-4 所示。



图 12-4　在台式机中的执行效果

如果在 Android 模拟器中运行上述程序，效果如图 12-5 所示。



图 12-5　在 Android 模拟器中的运行效果

对于上述代码来说，无论使用的是什么浏览器，运行效果并无太大区别。这是因为上述模板符合 HTML 5 语法标准，并且包含了 jQuery Mobile 的特定属性和 asset 文件（CSS、js）。

12.4　使用多页面模板

实例 135	在 Android 中使用多页面模板
源码路径	光盘:\daima\135
视频路径	光盘:\视频\135
实例必备	135.设置内部页面的页面标题.pdf

12.4.1　实例说明

jQuery Mobile 支持在一个 HTML 文档中嵌入多个页面，该策略可以用来预先获取最前面的多个页面，当载入子页面时，其响应时间会缩短。读者在下面的实例中可以看到，多页面文档与前面看到的单页面文档相同，第二个页面附加在第一个页面后面的情况除外。

12.4.2　具体实现

实例文件 duo.html 的具体代码如下所示。

```
<!DOCTYPE html>
<html>
	<head>
	<meta charset="utf-8">
	<title>Multi Page Example</title>
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
	<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
	<script type="text/javascript">/* Shared scripts for all internal and ajax-loaded pages */</script>
	<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
	</head>
<body>
<!-- First Page -->
<div data-role="page" id="home" data-title="Welcome">
	<div data-role="header">
		<h1>Multi-Page</h1>
	</div>
	<div data-role="content">
		<a href="#contact-info" data-role="button">联系我们</a>
	</div>
	<script type="text/javascript">
		/*Page specific scripts here.*/
	</script>
</div>
<!-- Second Page -->
<div data-role="page" id="contact-info" data-title="Contacts">
	<div data-role="header">
		<h1>联系我们</h1>
	</div>
	<div data-role="content">
		联系信息详情...
	</div>
</div>
</body>
</html>
```

上述代码在 Android 中的初始执行效果如图 12-6 所示。

单击“联系我们”按钮后会显示一个新界面，如图 12-7 所示。此新界面效果也是由上述代码实现的。



图 12-6　初始执行效果



图 12-7　显示一个新界面

12.5　使用 Ajax 驱动导航

实例 136	在 Android 中使用 Ajax 驱动导航
源码路径	光盘:\daima\136
视频路径	光盘:\视频\136
实例必备	136.分析 jQuery Mobile 的处理流程.pdf

12.5.1　实例说明

当一个单页面转换到另外一个单页面时，导航模型是不同的。例如，可以从多页面中提取出 contact 页面，然后命名为 contact.html 文件。现在的 home 页面（hijax.html）可以通过一个普通的 HTTP 链接引用返回到 contact 页面。

12.5.2　具体实现

实例文件 ajax.html 的具体代码如下所示。

```
<!DOCTYPE html>
<html>
        <head>
        <meta charset="utf-8">
        <title>Hijax Example</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
        <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
</head>
<body>
<!-- First Page -->
<div data-role="page">
        <div data-role="header">
                <h1>Ajax 页面</h1>
        </div>
        <div data-role="content">
                <a href="contact.html" data-role="button">联系我们</a>
        </div>
</div>
</body>
</html>
```

上述代码在 Android 中的初始执行效果如图 12-8 所示。

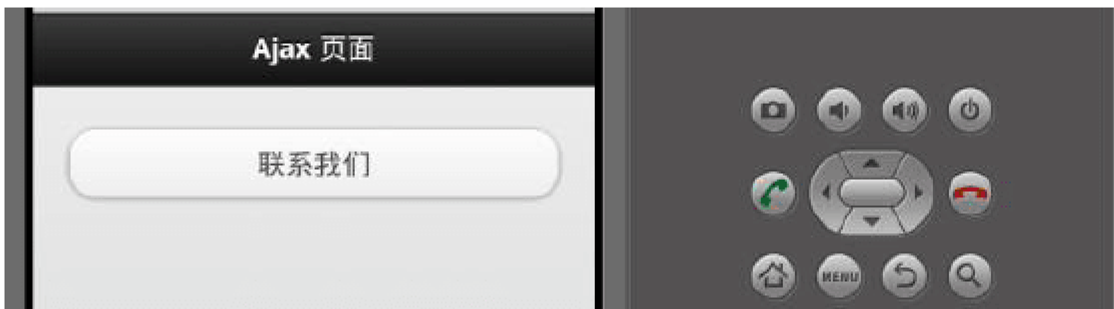


图 12-8　初始执行效果

单击“联系我们”按钮会跳转到新页面 contact.html，此文件的实现代码如下所示。

```
<div data-role="page">
        <div data-role="header">
              <h1>联系我们</h1>
        </div>

        <div data-role="content">
              电话：010-111111111</div>
        <div data-role="content">
              邮箱：7291017304@qq.com</div>
        <div data-role="content">地址：中国山东</div>
</div>
```

单击“联系我们”按钮会显示一个 Ajax 特效，如图 12-9 所示，然后显示如图 12-10 所示的新页面。



图 12-9　Ajax 特效导航



图 12-10　新页面效果

12.6　实现基本对话框效果

实例 137	在 Android 系统中实现对话框效果
源码路径	光盘:\daima\137
视频路径	光盘:\视频\137
实例必备	137.使用操作表.pdf

12.6.1　实例说明

对话框与页面相似，只不过对话框的边界是有间距的（inset），从而产生模态对话框（modal dialog）的外观。在对话框的设计方面，jQuery Mobile 相当灵活，可以创建确认对话框、警告对话框，甚至动作表单样式的对话框。

12.6.2 具体实现

实例文件 duihuakuang.html 的具体实现流程如下所示。

（1）实现链接级别的转换，具体代码如下所示。

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Multi Page Example</title>
		<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
	<style>
		.ui-header .ui-title, .ui-footer .ui-title { margin-right: 0 !important; margin-left: 0 !important; }
	</style>
	<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
	<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
	</head>
<body>

<!--第一页 -->
<div data-role="page" id="home">
	<div data-role="header">
		<h1>对话框实例</h1>
	</div>

	<div data-role="content">
		<a href="#terms" data-transition="slidedown">会员注册条款</a>
	</div>
</div>
```

（2）实现页面级别的转换，具体代码如下所示。

```
<!--第二页—对话框 -->
<div data-role="dialog" id="terms">
	<div data-role="header">
		<h1>注册条款</h1>
	</div>

	<div data-role="content" data-theme="c">
		你同意上述条款吗?
	<br><br>
		<a href="#home" data-role="button" data-inline="true" data-rel="back" data-theme="a">不同意！
</a><a href="javascript:agree();" data-role="button" data-inline="true">同意！ </a>
	</div>
```

（3）处理按钮进程，具体代码如下所示。

```
	<script>
		function agree() {
			//process dialog...
```

```
                //close dialog
                $('.ui-dialog').dialog('close');
            }
        </script>
</div>
</body>
</html>
```

本实例执行后的初始效果如图 12-11 所示。

单击“会员注册条款”超链接后显示如图 12-12 所示的对话框界面效果。

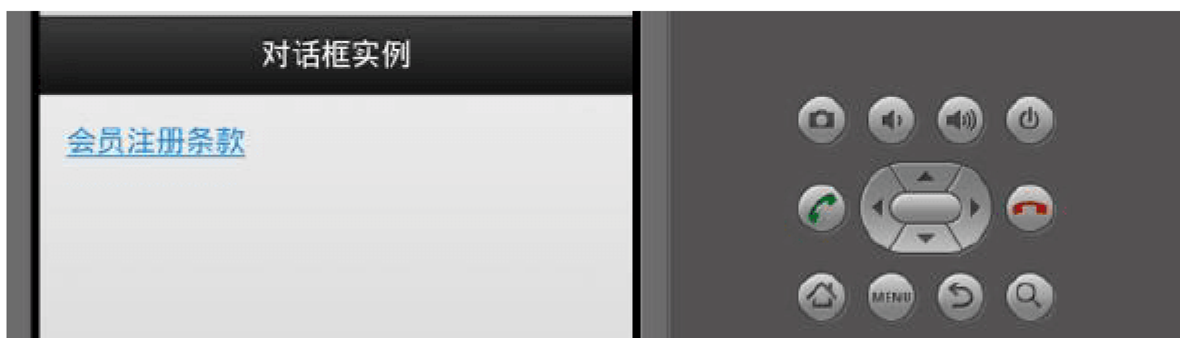


图 12-11　初始执行效果



图 12-12　对话框界面效果

12.7　实现竖屏和横屏自适应效果

实例 138	在 Android 系统中实现竖屏和横屏自适应效果
源码路径	光盘:\daima\138
视频路径	光盘:\视频\138
实例必备	138.Webkit 的媒体扩展.pdf

12.7.1　实例说明

在某些情况下，jQuery Mobile 将会创建响应式设计。下面讲解将 jQuery Mobile 的响应式设计应用于竖屏（pomait）模式和横屏（landscape）模式中的表单字段。例如，在竖屏视图中标签位于表单字段的上面。而当将设备横屏放置时表单字段和标签并排显示。这种响应式设计可以基于设备可用的屏幕真实状态提供最实用的体验。jQuery Mobile 为用户提供了很多这样优秀的 UX（用户体验）原则。

12.7.2　具体实现

实例文件 zishiyong.html 的具体实现代码如下所示。

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="utf-8">
    <title>Responsive Design Example</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
    <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
```

```
    <script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
</head>
<body>

<div data-role="page">
    <div data-role="header">
        <h1>会员注册</h1>
    </div>

    <div data-role="content">
        <label for="username">用户名:</label>
        <input type="text" name="username" id="username" value="" />

        <label for="password">密  码:</label>
        <input type="password" name="password" id="password" value="" />
    </div>
</div>
</body>
</html>
```

上述代码执行后的效果如图 12-13 所示，如果将设备纵向放置，则注册表单将自动旋转，实现自适应效果。



图 12-13　执行效果

12.8　实现全屏显示效果

实例 139	在 Android 系统中实现全屏显示效果
源码路径	光盘:\daima\139
视频路径	光盘:\视频\139
实例必备	139.可以用于定位页眉的 3 种样式.pdf

12.8.1　实例说明

页眉通常用于显示页面标题，还可以包含控件，以辅助用户在屏幕中进行导航或管理对象。页眉

栏显示当前屏幕的标题。此外，也可以在上面添加用于导航的按钮，或者是添加用来管理页面中的项目的控件。尽管页眉是可选的，但是通常用来提供活动页面的标题。

12.8.2　具体实现

实例文件 position-full.html 的具体实现代码如下所示。

```
<!DOCTYPE html>
<html>
	<head>
	<meta charset="utf-8">
	<title>Fullscreen Example</title>
	<meta name="viewport" content="width=device-width, maximum-scale=1">
	<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
	<style>
		.detailimage { width: 100%; text-align: center; margin-right: 0; margin-left: 0; }
		.detailimage img { width: 100%; }
	</style>
	<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
	<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
</head>
<body>
<div data-role="page" data-fullscreen="true">
	<div data-role="header" data-position="fixed">
		<h6>4/10</h6>
	</div>

	<div data-role="content">
		<div class="detailimage"><img src="images/123.jpg" /></div>
	</div>

	<!-- toolbar with icons -->
	<div data-role="footer" data-position="fixed">
		<div data-role="navbar">
			<ul>
				<li><a href="#" data-icon="forward"></a></li>
				<li><a href="#" data-icon="arrow-l"></a></li>
				<li><a href="#" data-icon="arrow-r"></a></li>
				<li><a href="#" data-icon="delete"></a></li>
			</ul>
		</div>
	</div>
</div>
</body>
</html>
```

执行上述代码后将首先显示一个有页眉的效果，如图 12-14 所示。

在如图 12-14 所示的效果中有一个用来显示照片全屏的页面，如果用户轻点屏幕，则页眉和页脚将会消失，这样便形成了一个全屏显示效果，如图 12-15 所示。再轻点屏幕页眉和页脚将会出现。

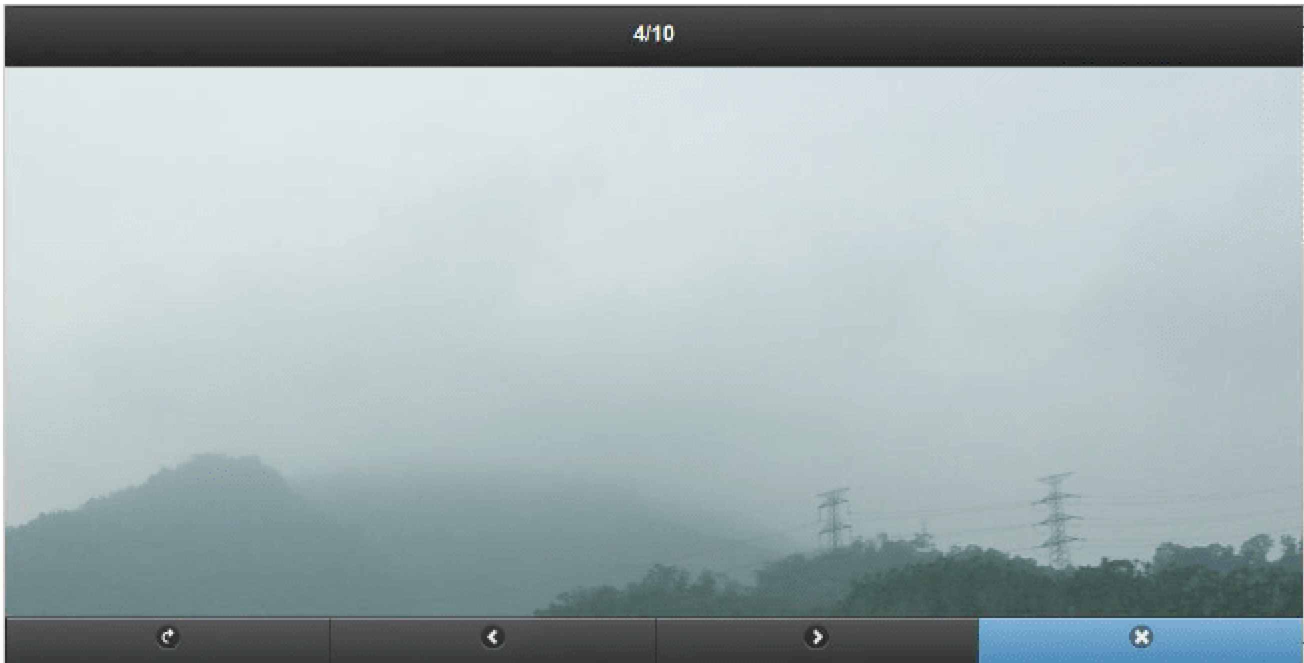


图 12-14　有页眉的效果



图 12-15　页眉消失后全屏显示

在本实例中有一个照片查看器，而且其页眉显示照片的计数信息，页脚显示一个工具栏以辅助导航、发送电子邮件或删除照片。

12.9　在表单中输入文本

实例 140	在 Android 系统的表单中输入文本
源码路径	光盘:\daima\140
视频路径	光盘:\视频\140
实例必备	140.将输入字段与其语义类型关联.pdf

12.9.1　实例说明

文本输入工作是移动设备上最麻烦的表单字段，当在物理或真实的 QWERTY 键盘上输入文字时，效率会非常低。所以在移动设备中，需要尽可能自动收集用户的信息。前面提到，设备 API 有助于简化这一用户体验。尽管最大限度地减少这些繁琐的任务是所期望的目标，但是有时必须使用文本输入来收集用户的反馈信息。

12.9.2　具体实现

实例文件 text.html 的具体实现代码如下所示。

```
<!DOCTYPE html>
<html>
        <head>
        <meta charset="utf-8">
        <title>Forms</title>
        <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-scale=1.0;">
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
        <style>
                label {
                    float: left;
                    width: 5em;
              }
                input.ui-input-text {
                display: inline !important;
                       width: 12em !important;
                }
                form p {
                        clear:left;
                        margin:1px;
                }
        </style>
        <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
</head>
<body>

<div data-role="page" data-theme="b">
        <div data-role="header">
                <h1>输入文本</h1>
        </div>

        <div data-role="content">
           <form id="test" id="test" action="#" method="post">
                <p style="margin-bottom:8px;">
                <label for="search" class="ui-hidden-accessible">Search</label>
        <input type="search" name="search" id="search" value="" placeholder="Search" data-theme="d" />
                </p>
                <p>
               <label for="text">名字:</label>
               <input type="text" name="text" id="text" value="" placeholder="Text" data-theme="d"/>
               </p>
                <p>
                <label for="number">编号:</label>
              <input type="number" name="number" id="number" value="" placeholder="Number" data-theme="d" />
                </p>
```

```
            <p>
          <label for="email">邮箱:</label>
          <input type="email" name="email" id="email" value="" placeholder="Email" data-theme="d" />
            </p>
            <p>
          <label for="url">网址:</label>
          <input type="url" name="url" id="url" value="" placeholder="URL" data-theme="d" />
            </p>
            <p>
          <label for="tel">电话:</label>
          <input type="tel" name="tel" id="tel" value="" placeholder="Telephone" data-theme="d" />
            </p>

            <!-- Future: http://www.w3.org/2011/02/mobile-web-app-state.html -->
          <!--
            <p>
          <label for="date">date:</label>
          <input type="date" name="date" id="date" value="" placeholder="Date" data-theme="d" />
          <p>
            -->


            <p>
            <label for="textarea">留言:</label>
            <textarea cols="40" rows="8" name="textarea" id="textarea" placeholder="Textarea" data-theme=
"d"></textarea>
            </p>
          </form>
        </div>
</div>

</body>
</html>
```

在上述实例代码中，通过为输入元素添加属性 data-theme 的方法，为文本选择一个合适的主题，从而增强表单字段的对比。执行后，如果在“名字”文本框中输入信息，则自动弹出文字键盘，如图 12-16 所示。如果在“编号”文本框中输入信息，则自动弹出数字键盘，如图 12-17 所示。



图 12-16　自动弹出文字键盘



图 12-17　自动弹出数字键盘

12.10　动态输入文本

实例 141	使用 textinput 插件动态输入文本
源码路径	光盘:\daima\141
视频路径	光盘:\视频\141
实例必备	141.使用选择菜单.pdf

12.10.1　实例说明

在 jQuery Mobile 应用中，可以给 input 元素直接绑定事件，可使用 jQuery Mobile 的虚拟事件，或者绑定 JavaScript 的标准事件，如 change、focus 和 blur 等。例如：

```
$( ".selector" ).bind( "change", function(event, ui) {
  ...
});
```

12.10.2　具体实现

实例文件 dynamic-text.html 的具体实现代码如下所示。

```
<div data-role="page" data-theme="b">
        <div data-role="header">
                <h1>动态输入文本</h1>
        </div>

        <div data-role="content">
           <form id="test" action="#" method="post">
                <a href="#" data-role="button" id="create-text1">创建文本输入框 1</a>
                <a href="#" data-role="button" id="create-text2">创建文本输入框 2</a>
                <br><br>
                <a href="#" data-role="button" id="disable-text1" data-theme="a">不可用输入框 1</a>
                <a href="#" data-role="button" id="enable-text1" data-theme="a">可用输入框 1</a>
           </form>
        </div>
                <script type="text/javascript">
                $( "#create-text1" ).bind( "click", function() {
                        $( '<input type="text" name="text1" id="text1" value="" placeholder="text1" data-theme="c" />' )
                                .insertAfter( "#create-text1" )
                                .textinput();
                });

                $( "#create-text2" ).bind( "click", function() {
                        $( '<input type="text" name="text2" id="text2" value="" placeholder="text2" />' )
                                .insertAfter( "#create-text2" )
                                .textinput({
                                        theme: 'c',
```

```
                    create: function(event) {
                        console.log( "Creating text input..." );
                        for (prop in event) {
                            console.log(prop + ' = ' + event[prop]);
                        }
                    }
                });
        });

        $( "#disable-text1" ).bind( "click", function() {
            $( "#text1").textinput( "disable" );
        });

        $( "#enable-text1" ).bind( "click", function() {
            $( "#text1" ).textinput( "enable" );
        });
        </script>
</div>
```

执行后的初始效果如图 12-18 所示。触摸按下某个按钮后会自动创建一个文本输入框，例如，触摸按下“创建文本输入框 1”按钮后会创建一个如图 12-19 所示的输入框。



图 12-18　初始效果



图 12-19　自动创建一个文本输入框

12.11　实现内置列表效果

实例 142	在 Android 中使用内置列表
源码路径	光盘:\daima\142
视频路径	光盘:\视频\142
实例必备	142.使用列表分割线.pdf

12.11.1　实例说明

在 jQuery Mobile 应用中，显示内置列表（inset list）时不会占据整个屏幕。相反，会自动存在于带有圆角的区域块内部，而且具有额外空间的边距设置。要创建一个内置列表，需要为列表元素添加

data-inset="true"属性。如果列表需要嵌入在有其他内容的页面中，内嵌列表会将列表设为边缘圆角，周围留有 magin 的块级元素。

12.11.2　具体实现

实例文件 inset.html 的具体实现代码如下所示。

```
<div data-role="page" data-add-back-btn="true">
	<div data-role="header">
		<h1>联系亲们</h1>
	</div>

	<div data-role="content">
		<ul data-role="listview" data-inset="true">
			<li data-role="list-divider">选择联系方式</li>
			<li><a href="#"><img src="images/75-phone.png" alt="Call" class="ui-li-icon">电话</a></li>
			<li><a href="#"><img src="images/112-envelope.png" alt="Email" class="ui-li-icon">邮件</a> </li>
			<li><a href="#"><img src="images/012-chat-2.png" alt="SMS" class="ui-li-icon">短信</a></li>
			<li><a  href="#"><img  src="images/1012-map.png"  alt="Directions"  class="ui-li-icon">腹语术</a>
</li>
		</ul>
	</div>
</div>
```

上述代码的执行效果如图 12-20 所示。



图 12-20　执行效果

12.12　开发一个 Web 版的电话簿系统

实例 143	在 Android 系统中开发一个 Web 版的电话簿系统
源码路径	光盘:\daima\143
视频路径	光盘:\视频\143
实例必备	143.使用页面初始化事件 Page initialization events.pdf

12.12.1　实例说明

本实例使用 HTML 5 和 PhoneGap 实现了一个经典的电话本工具，能够实现对设备内联系人信息的

管理，包括添加新信息、删除信息、快速搜索信息、修改信息、更新信息等功能。

12.12.2　具体实现

主页文件 main.html 的具体实现代码如下所示。

```
        <script src="./js/jquery.js"></script>
        <script src="./js/jquery.mobile-1.2.0.js"></script>
        <script src="./cordova-2.1.0.js"></script>

</head>
<body>
        <!-- Home -->
        <div data-role="page" id="page1" style="background-image: url(./img/bg.gif);" >
            <div data-theme="e" data-role="header">
                    <h2>电话本管理中心</h2>
            </div>
            <div data-role="content" style="padding-top:200px;">
                <a data-role="button" data-theme="e" href="./select.html" id="chaxun" data-icon="search" data-
iconpos= "left" data-transition="flip">查询</a>
                <a data-role="button" data-theme="e" href="./set.html" id="guanli"  data-icon="gear" data-iconpos=
"left"> 管理 </a>
            </div>
            <div data-theme="e" data-role="footer" data-position="fixed">
                <span class="ui-title">免费组织制作 v1.0</span>
            </div>

             <script type="text/javascript">
                      //App custom javascript
                sessionStorage.setItem("uid","");

                 $('#page1').bind('pageshow',function(){
                    $.mobile.page.prototype.options.domCache = false;

                  });
                 //等待加载 PhoneGap
                 document.addEventListener("deviceready", onDeviceReady, false);

                 // PhoneGap 加载完毕
                 function onDeviceReady() {
                  var db = window.openDatabase("Database", "1.0", "PhoneGap myuser", 200000);
                  db.transaction(populateDB, errorCB);
                 }
                   //填充数据库
                  function populateDB(tx) {
                       tx.executeSql('CREATE TABLE IF NOT EXISTS 'myuser' ('user_id' integer primary
key autoincrement , 'user_name' VARCHAR( 25 ) NOT NULL , 'user_phone' varchar( 15 ) NOT NULL , 'user_qq'
varchar( 15 ) , 'user_email' VARCHAR( 50 ), 'user_bz' TEXT)');

                  }

                  //事务执行出错后调用的回调函数
```

```
                    function errorCB(tx, err) {
                         alert("Error processing SQL: "+err);
                    }

                </script>
        </div>
    </body>
```

添加信息文件 add.html 的主要实现代码如下所示。

```
<script type="text/javascript" src="./js/jquery.js"></script>
</head>
<body>
 <!-- Home -->
      <div data-role="page" id="page1">
          <div data-theme="e" data-role="header">
              <a data-role="button" id="tjlxr" data-theme="e" data-icon="info" data-iconpos="right" class="ui-
btn-right">保存</a>
              <h3>添加联系人 </h3>
              <a data-role="button" id="czlxr" data-theme="e"   data-icon="refresh" data-iconpos="left" class=
"ui-btn-left"> 重置</a>
          </div>
          <div data-role="content">
              <form action="" data-theme="e" >
                  <div data-role="fieldcontain">
                      <fieldset data-role="controlgroup" data-mini="true">
                          <label for="textinput1">姓名：<input name="" id="textinput1" placeholder="联系人
姓名" value="" type="text" /></label>
                      </fieldset>
                      <fieldset data-role="controlgroup" data-mini="true">
                          <label for="textinput2">电话： <input name="" id="textinput2" placeholder="联系人
电话" value="" type="tel" /></label>
                      </fieldset>
                      <fieldset data-role="controlgroup" data-mini="true">
                          <label for="textinput3">QQ：<input name="" id="textinput3" placeholder="" value=" "
type="number" /></label>
                      </fieldset>
                       <fieldset data-role="controlgroup" data-mini="true">
                           <label for="textinput4">Email：<input name="" id="textinput4" placeholder="" value=" "
type="email" /></label>
                       </fieldset>
                       <fieldset data-role="controlgroup">
                           <label for="textarea1"> 备注： </label>
                           <textarea name="" id="textarea1" placeholder="" data-mini="true"></textarea>
                       </fieldset>
                  </div>
                  <div>
                      <a data-role="button"   id="back" data-theme="e" >返回</a>
                  </div>
              </form>

          </div>
          <script type="text/javascript">
```

```
                $.mobile.page.prototype.options.domCache = false;
                var textinput1 = "";
                var textinput2 = "";
                var textinput3 = "";
                var textinput4 = "";
                var textarea1 = "";
                  $("#tjlxr").click(function(){

                    textinput1 = $("#textinput1").val();
                    textinput2 = $("#textinput2").val();
                    textinput3 = $("#textinput3").val();
                    textinput4 = $("#textinput4").val();
                    textarea1 = $("#textarea1").val();
                   var db = window.openDatabase("Database", "1.0", "PhoneGap myuser", 200000);
                   db.transaction(addBD, errorCB);
                  });

                  function addBD(tx){
                    tx.executeSql("INSERT  INTO  'myuser'  ('user_name','user_phone','user_qq',  'user_email','user_
bz') VALUES ('"+textinput1+"','"+textinput2+"','"+textinput3+"','"+textinput4+"','"+textarea1+"')", [ ], successCB, errorCB);
                  }
                  $("#czlxr").click(function(){
                    $("#textinput1").val("");
                    $("#textinput2").val("");
                    $("#textinput3").val("");
                    $("#textinput4").val("");
                    $("#textarea1").val("");
                  });
                  $("#back").click(function(){
                    successCB();
                  });
                  //等待加载 PhoneGap
                  document.addEventListener("deviceready", onDeviceReady, false);

                  //PhoneGap 加载完毕
                  function onDeviceReady() {
                   var db = window.openDatabase("Database", "1.0", "PhoneGap myuser", 200000);
                   db.transaction(populateDB, errorCB);
                  }
                   //填充数据库
                   function populateDB(tx) {
                         //tx.executeSql('DROP TABLE IF EXISTS 'myuser'');
                         tx.executeSql('CREATE  TABLE  IF  NOT  EXISTS  'myuser'  ('user_id'  integer  primary  key
autoincrement , 'user_name' VARCHAR( 25 ) NOT NULL , 'user_phone' varchar( 15 ) NOT NULL , 'user_qq'
varchar( 15 ) , 'user_email' VARCHAR( 50 ), 'user_bz' TEXT)');
                        //tx.executeSql("INSERT INTO 'myuser' ('user_name', 'user_phone', 'user_qq', 'user_email',
'user_bz') VALUES ('刘',12222222,222,'nlllllull','null')");
                        //tx.executeSql("INSERT INTO 'myuser' ('user_name', 'user_phone', 'user_qq', 'user_email',
'user_bz') VALUES ('张山',12222222,222,'nlllllull','null')");
                        //tx.executeSql("INSERT INTO 'myuser' ('user_name', 'user_phone', 'user_qq', 'user_email',
'user_bz') VALUES ('李四',12222222,222,'nlllllull','null')");
```

```
					//tx.executeSql("INSERT INTO 'myuser' ('user_name', 'user_phone', 'user_qq', 'user_email', 'user_bz') VALUES ('李四搜索',12222222,222,'nllllull','null')");
					//tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');
				}

				//事务执行出错后调用的回调函数
				function errorCB(tx, err) {
					alert("Error processing SQL: "+err);
				}

				//事务执行成功后调用的回调函数
				function successCB() {
					$.mobile.changePage ('set.html', 'fade', false, false);
				}
			</script>
		</div>
</body>
```

信息修改文件 modifiry.html 的主要实现代码如下所示。

```
		<div data-role="page" id="page1">
			<div data-theme="e" data-role="header">
				<a data-role="button"   id="tjlxr" data-theme="e" data-icon="info" data-iconpos="right" class="ui-btn-right">修改</a>
				<h3>修改联系人 </h3>
				<a data-role="button" id="back" data-theme="e" data-icon="refresh" data-iconpos="left" class="ui-btn-left"> 返回</a>
			</div>
			<div data-role="content">
				<form action="" data-theme="e" >
					<div data-role="fieldcontain">
						<fieldset data-role="controlgroup" data-mini="true">
							<label for="textinput1">姓名：<input name="" id="textinput1" placeholder="联系人姓名" value="" type="text" /></label>
						</fieldset>
						<fieldset data-role="controlgroup" data-mini="true">
							<label for="textinput2">电话： <input name="" id="textinput2" placeholder="联系人电话" value="" type="tel" /></label>
						</fieldset>
						<fieldset data-role="controlgroup" data-mini="true">
							<label for="textinput3">QQ：<input name="" id="textinput3" placeholder="" value=" " type="number" /></label>
						</fieldset>
						 <fieldset data-role="controlgroup" data-mini="true">
							<label for="textinput4">Email：<input name="" id="textinput4" placeholder="" value=" " type="email" /></label>
						 </fieldset>
						 <fieldset data-role="controlgroup">
							<label for="textarea1"> 备注： </label>
							<textarea name="" id="textarea1" placeholder="" data-mini="true"></textarea>
						 </fieldset>
					</div>
				</form>
```

```
            </div>
            <script type="text/javascript">
            $.mobile.page.prototype.options.domCache = false;
             var textinput1 = "";
              var textinput2 = "";
              var textinput3 = "";
              var textinput4 = "";
              var textarea1 = "";
              var uid = sessionStorage.getItem("uid");
              //==========================================================================
              $("#tjlxr").click(function(){

                  textinput1 = $("#textinput1").val();
                  textinput2 = $("#textinput2").val();
                  textinput3 = $("#textinput3").val();
                  textinput4 = $("#textinput4").val();
                  textarea1 = $("#textarea1").val();
                var db = window.openDatabase("Database", "1.0", "PhoneGap myuser", 200000);
                  db.transaction(modfiyBD, errorCB);
             });
                function modfiyBD(tx){
                 // alert("UPDATE 'myuser'SET 'user_name'='"+textinput1+"', 'user_phone'="+textinput2+",'user_qq'=
"+textinput3
                        //+",'user_email'='"+textinput4+"', 'user_bz'='"+textarea1+"' WHERE userid="+uid);
                  tx.executeSql("UPDATE  'myuser'SET  'user_name'='"+textinput1+"',  'user_phone'="+textinput2+",
'user_ qq'="+textinput3+",'user_email'='"+textinput4+"', 'user_bz'='"+textarea1+"' WHERE user_id="+uid, [ ], successCB,
errorCB);
                }
//==========================================================================
                $("#back").click(function(){
                  successCB();
                });


                //====================================================

                document.addEventListener("deviceready", onDeviceReady, false);
                // PhoneGap 加载完毕
                function onDeviceReady() {
                 var db = window.openDatabase("Database", "1.0", "PhoneGap myuser", 200000);
                 db.transaction(selectDB, errorCB);
                }
                 function selectDB(tx) {
                      //alert("SELECT * FROM myuser where user_id="+uid);
                    tx.executeSql("SELECT * FROM myuser where user_id="+uid, [ ], querySuccess, errorCB);
                 }
                 //事务执行出错后调用的回调函数
                 function errorCB(tx, err) {
                      alert("Error processing SQL: "+err);
                 }
                 //事务执行成功后调用的回调函数
                 function successCB() {
```

```
                    $.mobile.changePage ('set.html', 'fade', false, false);
                }
                function querySuccess(tx, results) {
                    var len = results.rows.length;
                    for (var i=0; i<len; i++){
                    //写入到 logcat 文件
                    //console.log("Row = " + i + " ID = " + results.rows.item(i).user_id + " Data = " +
results.rows.item(i).user_name);
                     $("#textinput1").val(results.rows.item(i).user_name);
                     $("#textinput2").val(results.rows.item(i).user_phone);
                     $("#textinput3").val(results.rows.item(i).user_qq);
                     $("#textinput4").val(results.rows.item(i).user_email);
                     $("#textarea1").val(results.rows.item(i).user_bz);
                }
            }
             </script>
        </div>
```

执行后的效果分别如图 12-21 和图 12-22 所示。



图 12-21　系统主界面



图 12-22　系统管理界面

12.13　搭建 PhoneGap 开发环境

实例 144	为 Android 系统搭建 PhoneGap 开发环境
源码路径	
视频路径	光盘:\视频\144
实例必备	144.PhoneGap 移动 Web 开发的步骤.pdf

12.13.1　实例说明

在使用 PhoneGap 进行移动 Web 开发之前，需要先搭建 PhoneGap 开发环境。在安装 PhoneGap 开发环境之前，需要先安装如下框架。

- ☑ Java SDK
- ☑ Eclipse
- ☑ Android SDK
- ☑ ADT Plugin

12.13.2 具体实现

在编写本书时，PhoneGap 的最新版本是 2.9.0。获得 PhoneGap 开发包的基本流程如下所示。

（1）登录 PhoneGap 的官方网站 http://phonegap.com/download/，如图 12-23 所示。

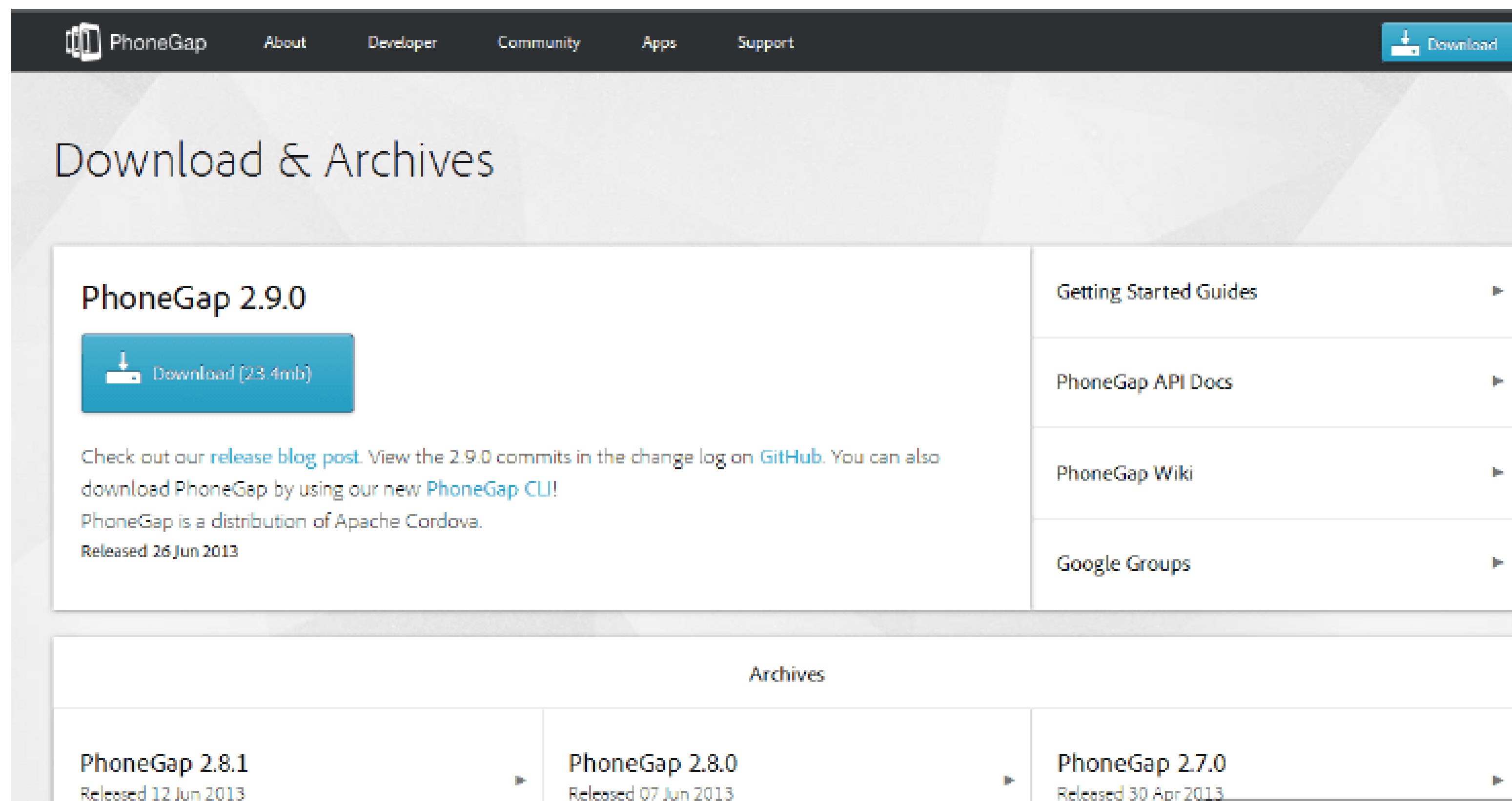


图 12-23　PhoneGap 的官方网站

（2）单击最新版本下方的按钮下载 PhoneGap 开发包，下载成功后的压缩包名为 phonegap-2.9.0.zip。

（3）解压缩文件 phonegap-2.9.0.zip，假设解压到本地硬盘的 D 目录下，解压后的根目录名是 phonegap-2.9.0，双击打开后的效果如图 12-24 所示。

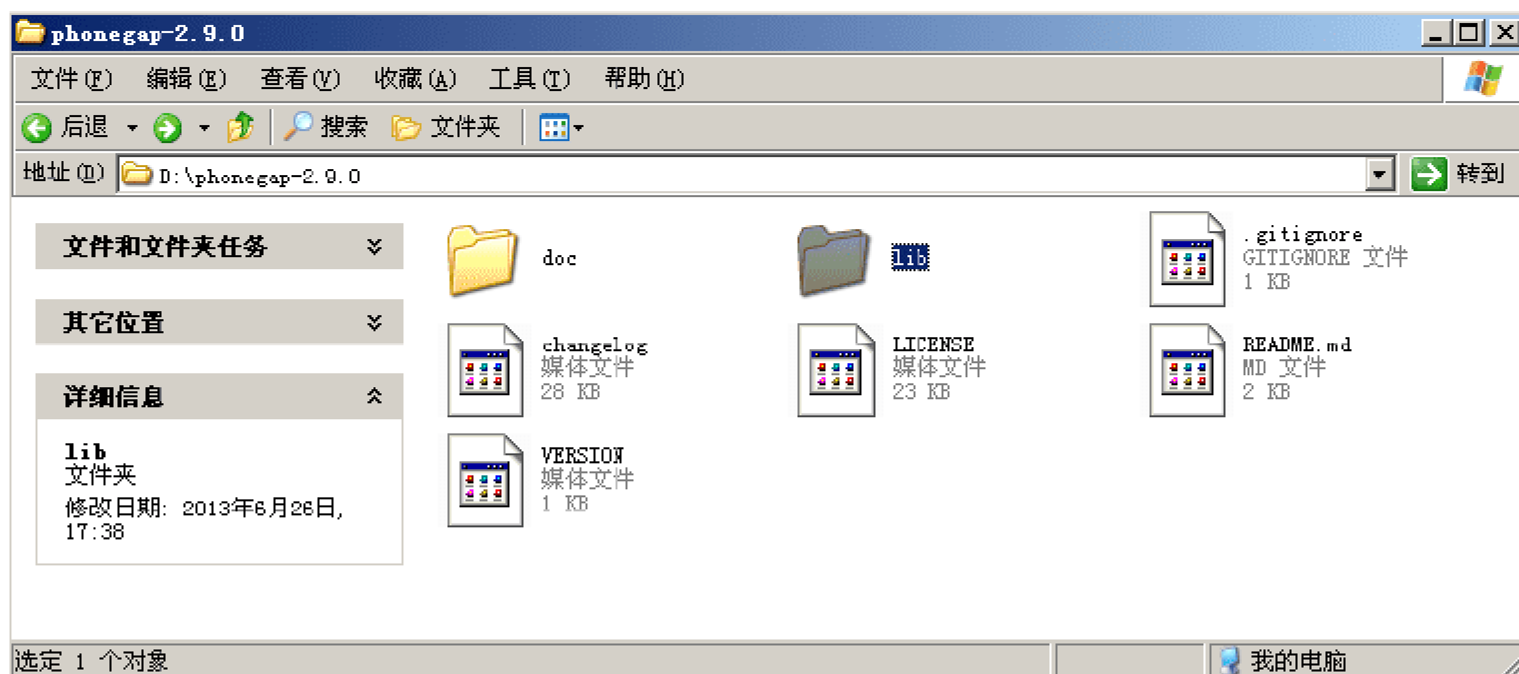


图 12-24　phonegap-2.9.0 的根目录

对图 12-24 中各个子目录的具体说明如下所示。

- ☑ doc：包含了 PhoneGap 的源代码文档，如图 12-25 所示。

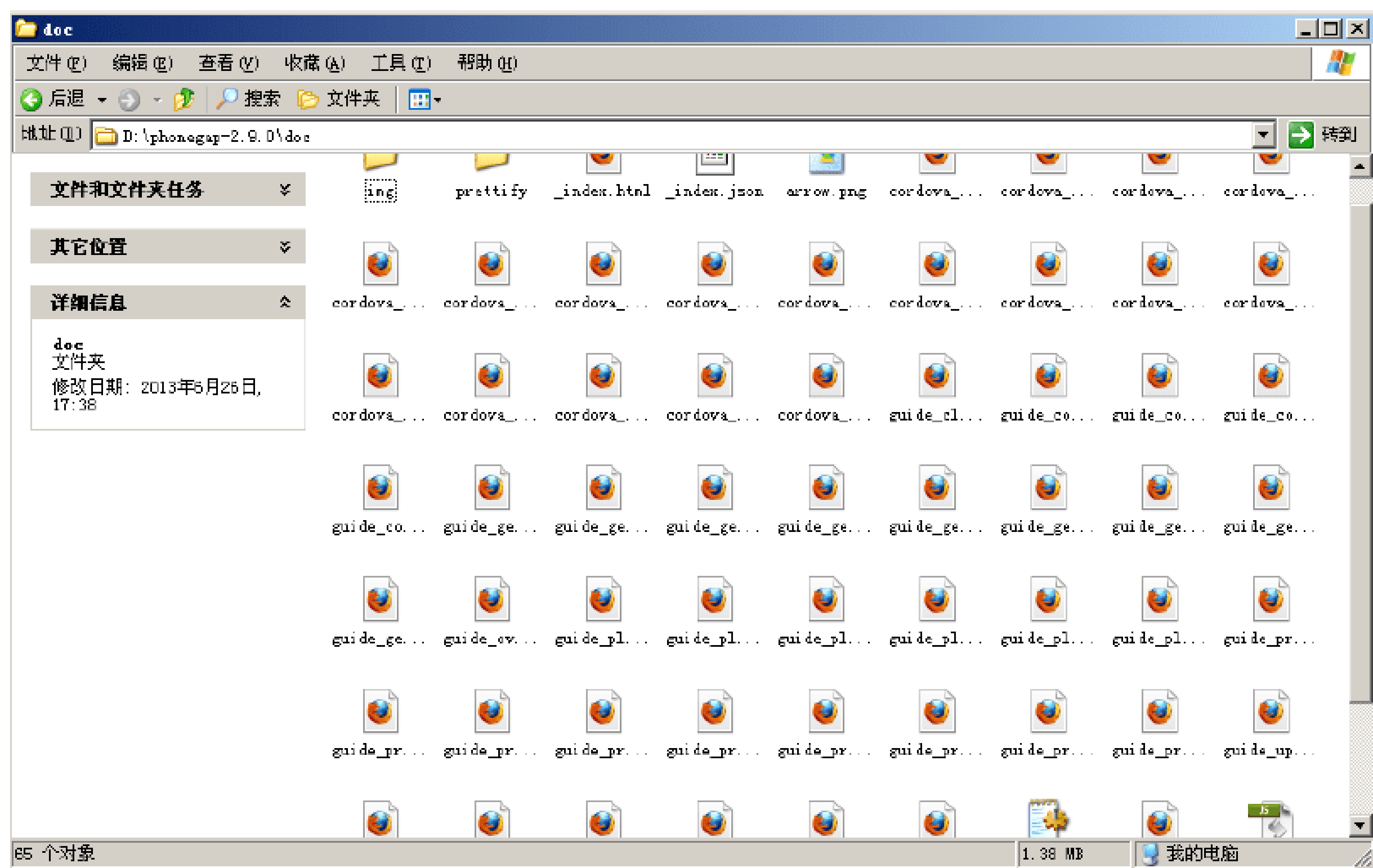


图 12-25　doc 目录

☑　lib：包含了 PhoneGap 支持的各种平台，如图 12-26 所示。

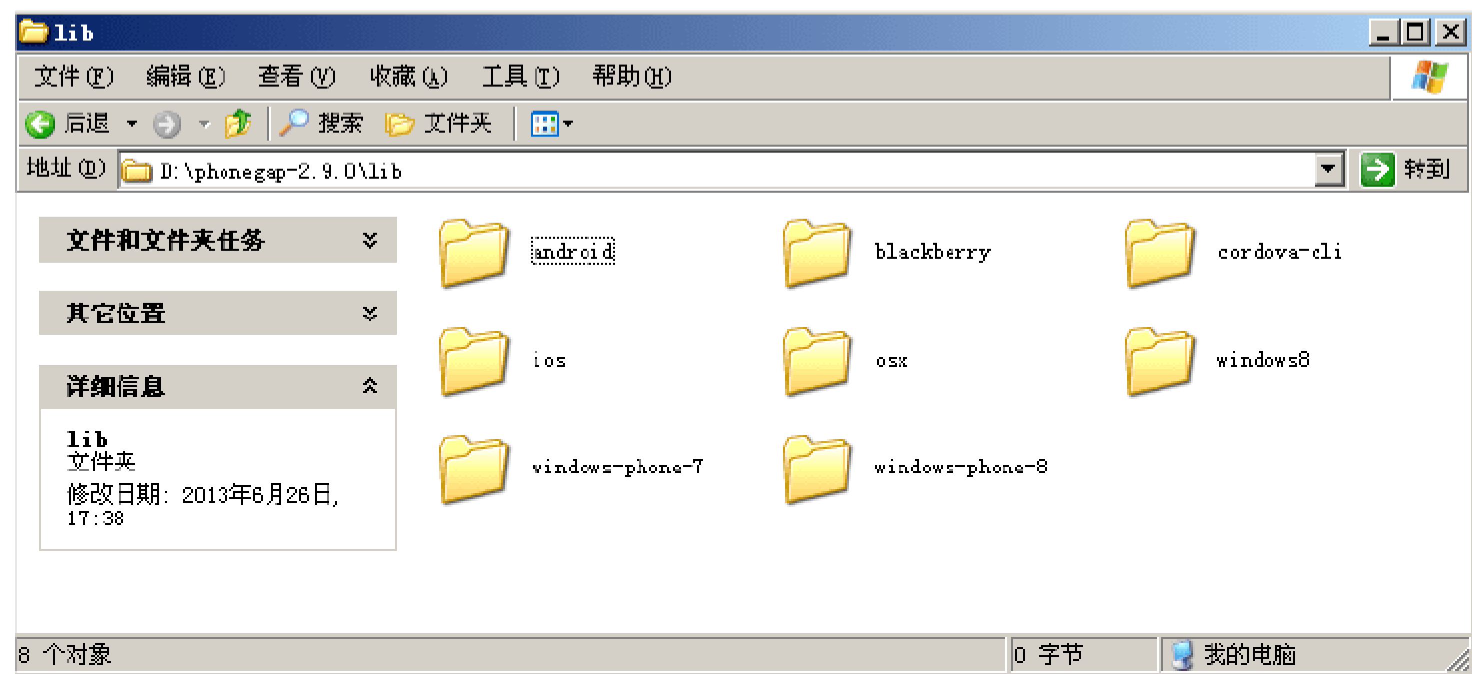


图 12-26　lib 目录

☑　changelog：一个日志文件，保存了更改历史记录信息和作者信息等。
☑　LICENSE：Apache 软件许可证（v2 版本）。
☑　VERSION：版本信息。
☑　README.md：帮助文档。

12.14　创建基于 PhoneGap 的 HelloWorld 程序

实例 145	在 Android 系统中创建基于 PhoneGap 的 HelloWorld 程序
源码路径	光盘:\daima\145
视频路径	光盘:\视频\145
实例必备	145.使用 PhoneGap API.pdf

12.14.1 实例说明

在本实例中将创建第一个 PhoneGap-Android 原生程序 HelloWorld。首先，利用 HTML、CSS 和 JavaScript 来搭建一个标准的 Web 应用程序，然后用 PhoneGap 封装来访问移动设备的基本信息，在 Android 模拟器上调试成功后，最后部署到实体机。为了在不同的设备上得到一样的渲染效果，将采用 jQuery Mobile 来设计应用程序界面。

12.14.2 具体实现

1. 首先建立一个基于 Web 的 Android 应用

创建标准 Android 应用的操作步骤如下所示。

（1）启动 Eclipse，选择 File | New | Other 命令，然后在向导的树形结构中找到 Android 节点。并单击 Android Project，在项目名称上填写 HelloWorld。

（2）单击 Next 按钮，选择目标 SDK，此处选择 Android 2.3.3。单击 Next 按钮，在其中填写包名 com.adobe.phonegap，如图 12-27 所示。

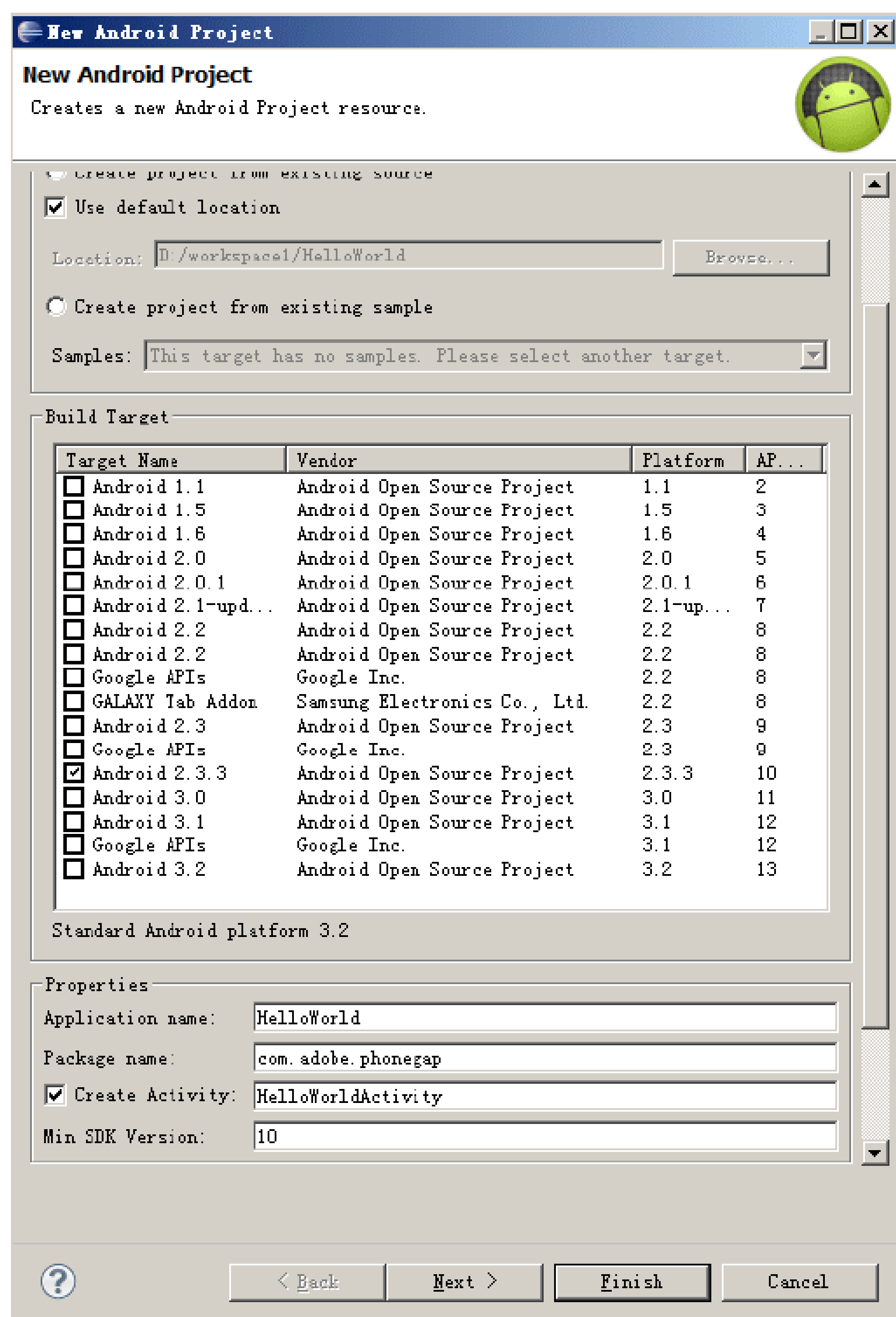


图 12-27 创建 Android 工程

（3）单击 Finish 按钮，此时将成功构建一个标准的 Android 项目。如图 12-28 所示展示了当前项目的目录结构。

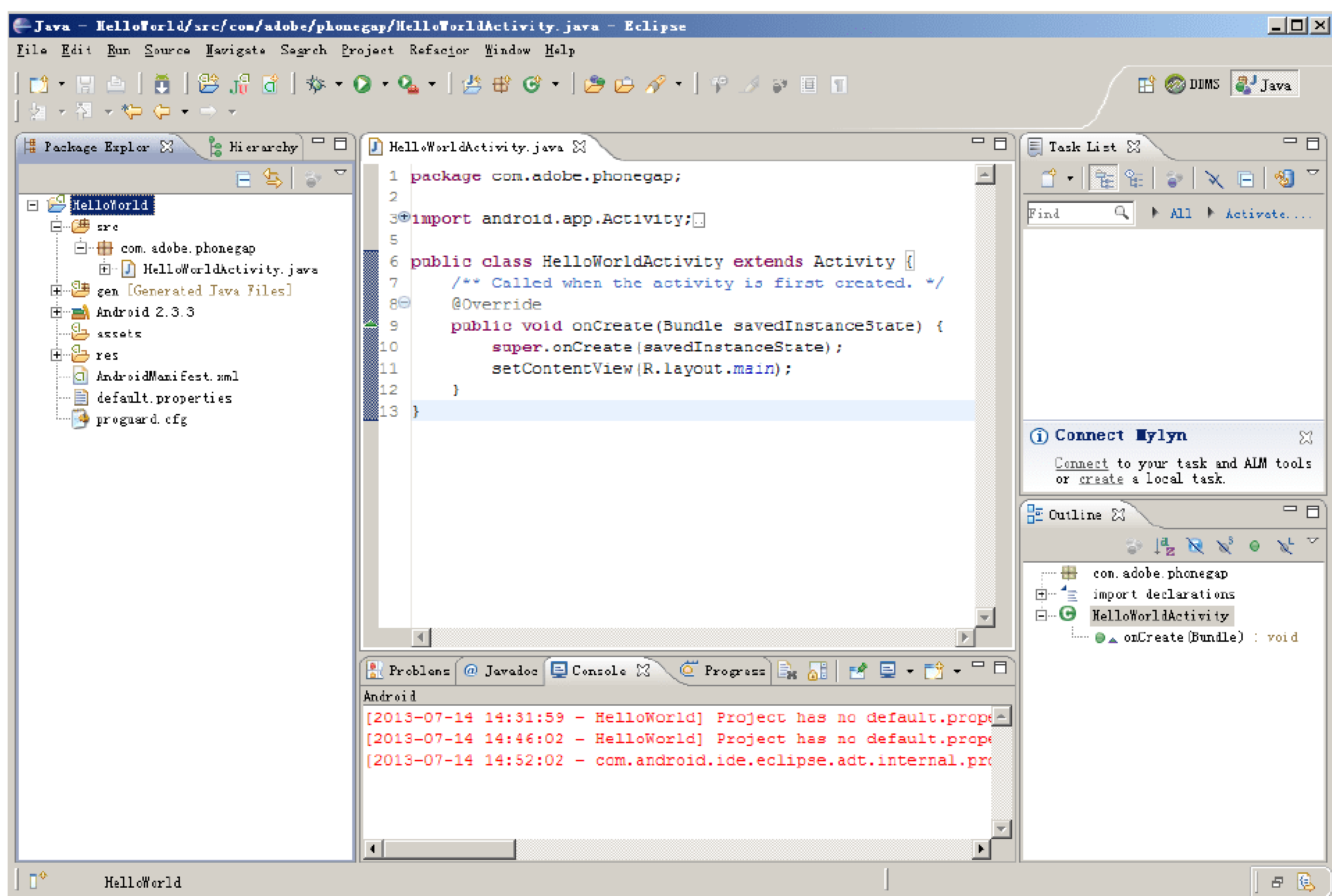


图 12-28　创建的 Android 工程

2. 添加 Web 内容

在 HelloWorld 中，将要添加的 Web 页面只有 index.html，该页面要完成的功能是在内容区域输出 HelloWorld。为了确保在不同的移动平台上显示一样的效果，此处使用 jQuery Mobile 来设计 UI。

（1）在 HelloWorld 程序的 assets 目录下创建 www 文件夹，这个文件夹是所有 Web 内容的容器。

（2）下载 jQuery Mobile，笔者在此实例使用的版本是 1.1.0 RC1。除了需要 jQuery Mobile 的 CSS 和相关 JavaScript 文件外，还需要用到 jquery.js。

（3）下载完 jQuery Mobile 并解压缩后，将 jquery.mobile-1.0.min.css、jquery.mobile-1.0.min.js 和 jquery.js 放置在 www 文件夹下，如图 12-29 所示。



图 12-29　添加 jQuery Mobile 文件

（4）开始编写文件 index.html，该页面是一个单页结构，共包含 3 部分，分别是页头、内容和页脚。文件 index.html 的具体代码如下所示。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>index.html</title>
    <link rel="stylesheet" href="jquery.mobile-1.0.1.min.css" />
    <script type="text/javascript" charset="utf-8" src="jquery.js"></script>
```

```
        <script type="text/javascript" charset="utf-8" src="jquery.mobile-1.0.1.min.js"></script>
</head>
<body>
<!-- begin first page -->
<div id="page1" data-role="page" >
<header data-role="header"><h1>Hello World</h1></header>
<div data-role="content" class="content">
<h3>设备信息</h3>

</ul>
</div>
<footer data-role="footer"><h1>Footer</h1></footer>
</div>
<!-- end first page -->
</body>
</html>
```

目前，该页面无法显示在移动设备中，在桌面浏览器上的显示效果如图 12-30 所示。

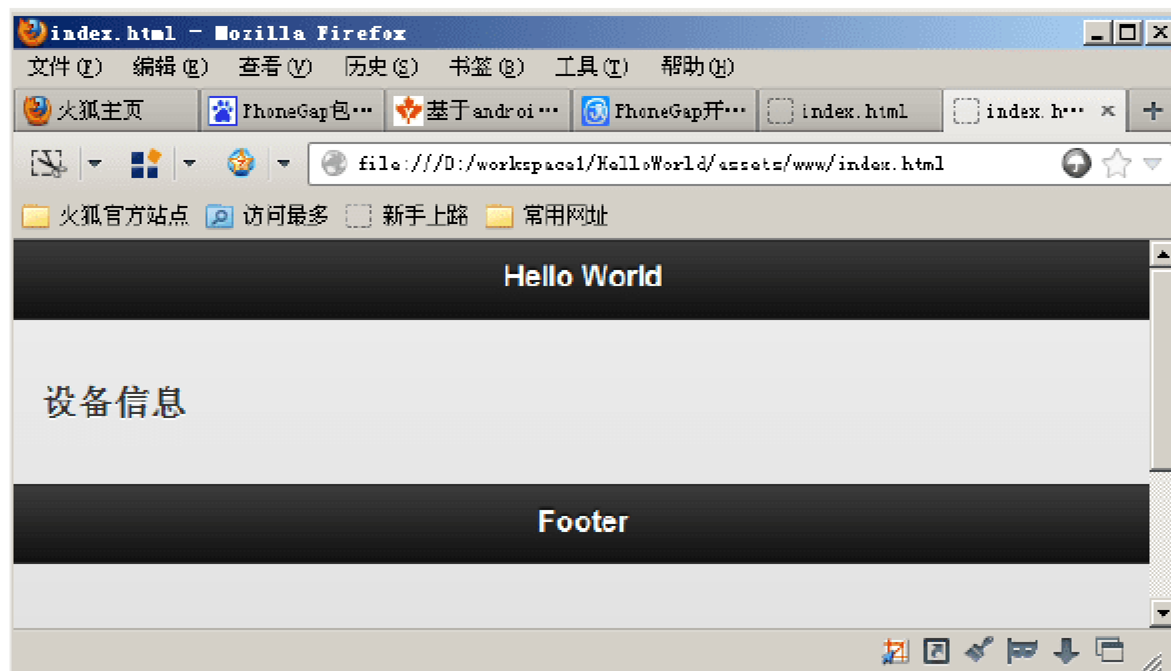


图 12-30　文件 index.html 的执行效果

3. 利用 PhoneGap 封装成移动 Web 应用

整个封装过程可以分为如下所示的 4 部分。

☑ 第 1 部分：修改项目结构，即创建一些必要的目录结构。

☑ 第 2 部分：引入 PhoneGap 相关文件，包含 cordova.js 和 cordova.jar，其中 cordova.js 主要用于 HTML 页面，而 cordova.jar 作为 Java 库文件引入。

☑ 第 3 部分：修改项目文件（包含 HTML 页面和 activity 类文件）。

☑ 第 4 部分：是可选的，就是修改项目元数据 AndroidManifest.xml，可以根据实际需要来修改该配置文件。

在接下来的内容中，将逐一介绍每一部分的具体实现过程。

（1）修改项目结构。在项目的根目录下创建 libs 和 assets\www 文件夹，前者是将要添加的 cordova.jar 包的容器，后者（该文件夹在“添加 Web 内容”中已经创建）是 Web 内容的容器。

（2）引入 PhoneGap 相关文件。在前面已经下载了最新的 PhoneGap 发布包 2.9.0。进入发布包的 \lib\android 目录，将文件 cordova.js 复制到 assets\www 目录下，将 cordova-2.9.0.jar 库文件复制到 libs 目录下，将 XML 文件夹复制到 res 目录下，作为 res 目录的一个子目录。在 PhoneGap 2.0 以前，XML 文件夹包含两个配置文件 cordova.xml 和 plugins.xml，从 2.0 开始，这两个文件合并成一个 config.xml。

修改项目的 Java 构建路径，把 libs 下的 cordova-2.9.0.jar 添加到编译路径中。

（3）修改项目文件。修改默认的 Java 文件 HelloWorldActivity，使其继承 DroidGap，修改后的代码如下所示。

```
package com.adobe.phonegap;
import org.apache.cordova.DroidGap;
import android.app.Activity;
import android.os.Bundle;
public class HelloWorldActivity extends DroidGap {
    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
    }
}
```

在上述代码中，DroidGap 是 PhoneGap 提供的，此类继承自 android.app.Activity 类。如果需要 PhoneGap 提供的 API 访问设备的原生功能或者设备信息，则需要在 index.html 的<header>标签中加入如下代码：

```
<script type="text/javascript" charset="utf-8" src="cordova.js" >
```

在本实例中，先试验一下不引入 cordova.js 时的情况，此时在模拟器上的运行效果如图 12-31 所示。

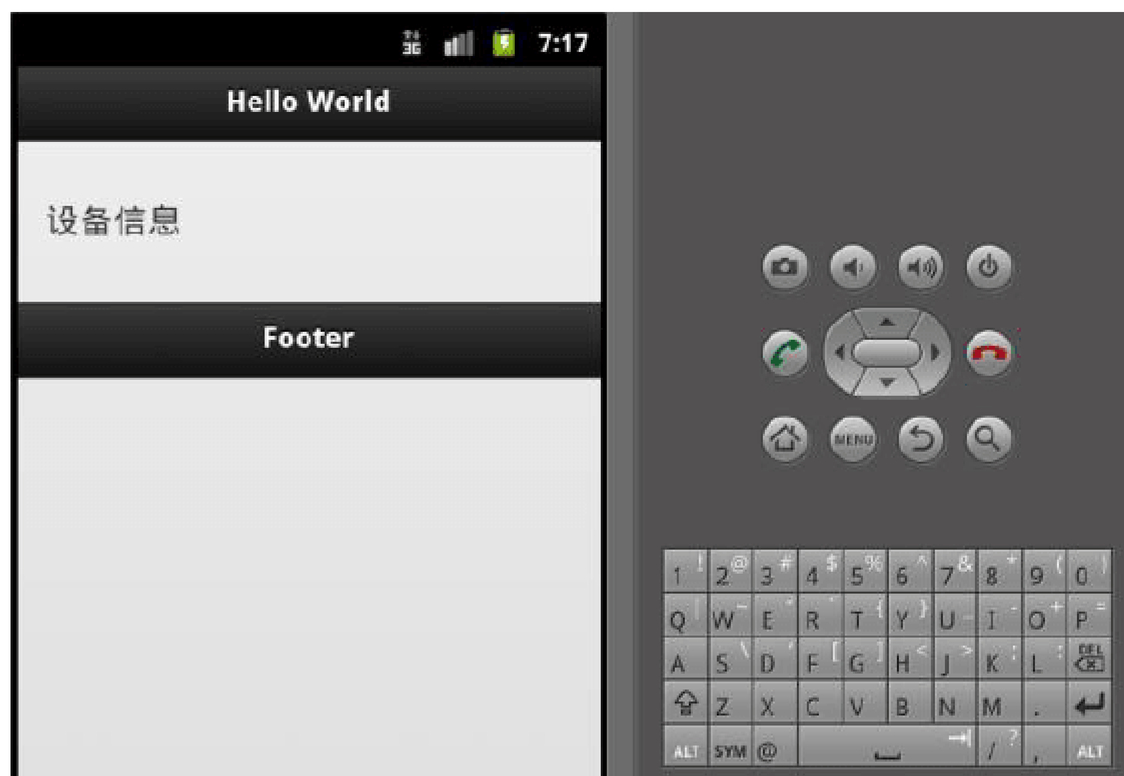


图 12-31　不引入 cordova.js 时的执行效果

现在修改文件 index.html，将文本 I am here 替换为“设备信息”。更改后的 index.html 页面的代码如下所示。

```
    <link rel="stylesheet" href="jquery.mobile-1.0.1.min.css" />
    <script type="text/javascript" charset="utf-8" src="jquery.js"></script>
    <script type="text/javascript" charset="utf-8" src="jquery.mobile-1.0.1.min.js"></script>
    <script type="text/javascript" charset="utf-8" src="cordova.js" ></script>
    <script type="text/javascript" charset="utf-8">

    $( function() {

    });
    $(document).ready(function(){
```

```
            console.log("jquery ready");
            document.addEventListener("deviceready", onDeviceReady, false);
            console.log("register the listener");
      });

      function onDeviceReady()
      {
            console.log("onDeviceReady");
            $(".content").html("<ul  data-role='listview'><li>"+device.name+"</li><li>"+device.cordova+"</li><li>"+
device.platform+"</li><li>"+device.version+"</li><li>"+device.uuid+"</li></ul>");
      }
      </script>
</head>
<body>
<!-- begin first page -->
<div id="page1" data-role="page" >
<header data-role="header"><h1>Hello World</h1></header>
<div data-role="content" class="content">
<h3>设备信息</h3>
</ul>
</div>
<footer data-role="footer"><h1>Footer</h1></footer>
</div>
```

在上述代码中，使用函数 onDeviceReady 调用$(".content").html 函数来修改 div 中的 HTML 内容。

4．修改权限文件 AndroidManifest.xml

在文件 AndroidManifest.xml 中，增加访问网络和照相机的权限，并添加适用不同分辨率的设置代码。文件 AndroidManifest.xml 的具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.adobe.phonegap"
        android:versionCode="1"
        android:versionName="1.0">

      <supports-screens   android:largeScreens="true"   android:normalScreens="true"   android:smallScreens=
"true"   android:resizeable="true" android:anyDensity="true"   />
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.VIBRATE" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.RECEIVE_SMS" />
  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
  <uses-permission android:name="android.permission.READ_CONTACTS" />
  <uses-permission android:name="android.permission.WRITE_CONTACTS" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

```
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
    <uses-sdk android:minSdkVersion="10" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloWorldActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

至此，整个实例介绍完毕，此时在 Android 中的执行效果如图 12-32 所示。

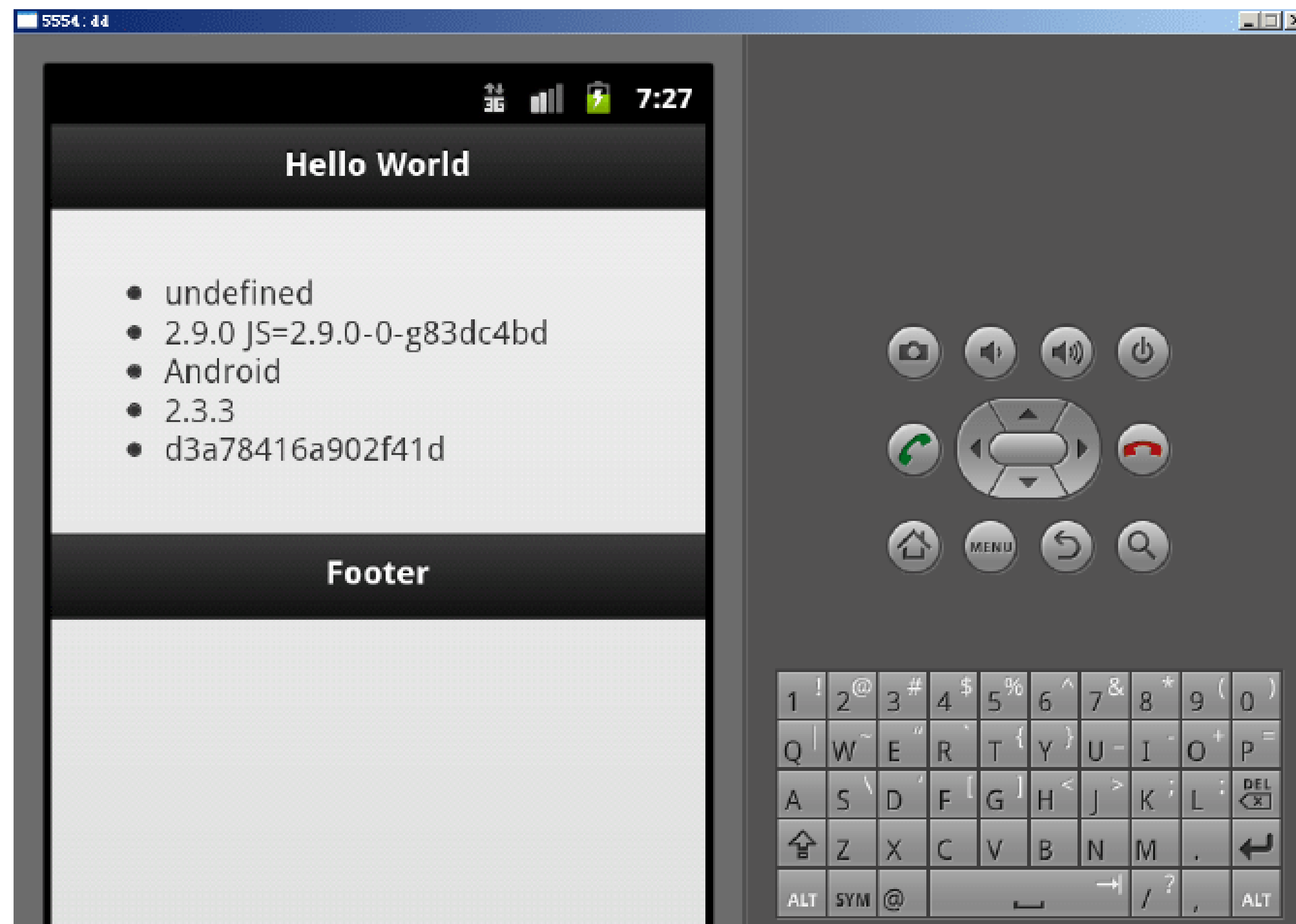


图 12-32　最终的执行效果

第 13 章　Google API 服务

谷歌公司为开发人员提供了很多 API 接口，通过这些接口可以实现谷歌应用功能，例如，GPS 定位和地图就是通过 Google API 实现的。Android 作为谷歌旗下的杰出产品，可以使用谷歌官方提供的很多强大的服务功能，例如，地图 API、日历 API、相册 API 和文件 API 等。本章将通过几个典型实例的实现过程，详细介绍 Android 系统中使用谷歌 API 服务的基本知识。

13.1　获取当前位置的坐标

实例 146	获取当前位置的坐标
源码路径	光盘:\daima\146
视频路径	光盘:\视频\146
实例必备	146.类 location 详解.pdf ① Google Map API ② Android Location API

13.1.1　实例说明

Android 支持 GPS 和网络地图，通常将各种不同的定位技术称为 LBS（Location Based Service，基于位置的服务），是通过电信移动运营商的无线电通信网络（如 GSM 网、CDMA 网）或外部定位方式（如 GPS）获取移动终端用户的位置信息（地理坐标或大地坐标），在 GIS（Geographic Information System，地理信息系统）平台的支持下，为用户提供相应服务的一种增值业务。在本实例中，使用 GPS 定位技术获取了当前位置的坐标信息。

13.1.2　具体实现

（1）编写文件 AndroidManifest.xml，在其中声明 ACCESS_FINE_LOCATION 权限，具体代码如下所示。

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

（2）编写主程序文件，在 onCreate(Bundle savedInstanceState)中获取当前位置信息，主要代码如下所示。

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        LocationManager locationManager;
        String serviceName = Context.LOCATION_SERVICE;
```

```
            locationManager = (LocationManager)getSystemService(serviceName);
            Criteria criteria = new Criteria();
            criteria.setAccuracy(Criteria.ACCURACY_FINE);
            criteria.setAltitudeRequired(false);
            criteria.setBearingRequired(false);
            criteria.setCostAllowed(true);
            criteria.setPowerRequirement(Criteria.POWER_LOW);
            String provider = locationManager.getBestProvider(criteria, true);
            Location location = locationManager.getLastKnownLocation(provider);
            updateWithNewLocation(location);
             /*每隔 1000ms 更新一次，并且不考虑位置的变化*/
            locationManager.requestLocationUpdates(provider, 2000, 10,
                    locationListener);
      }
```

在上述代码中，使用 LocationManager 周期获得当前设备的一个类。要获取 LocationManager 实例，需要调用方法 Context.getSystemService()并传入服务名 LOCATION_SERVICE("location")。在创建 LocationManager 实例后可以调用 getLastKnownLocation()方法将上一次 LocationManager 获得有效位置信息以 Location 对象的形式返回。

（3）定义方法 updateWithNewLocation(Location location)更新显示用户界面，主要代码如下所示。

```
private void updateWithNewLocation(Location location) {
 String latLongString;
 TextView myLocationText;
 myLocationText = (TextView)findViewById(R.id.myLocationText);
 if (location != null) {
 double lat = location.getLatitude();
 double lng = location.getLongitude();
 latLongString = "纬度:" + lat + "\n 经度:" + lng;
 } else {
 latLongString = "获取地理信息失败";
 }
 myLocationText.setText("当前坐标位置是:\n" +
 latLongString);
}
```

（4）定义 LocationListener 对象监听坐标改变时事件，如果 Provider 传进相同的坐标，就不会被触发，主要代码如下所示。

```
private final LocationListener locationListener = new LocationListener() {
  public void onLocationChanged(Location location) {
  updateWithNewLocation(location);
  }
  public void onProviderDisabled(String provider){
  updateWithNewLocation(null);
  }
  public void onProviderEnabled(String provider){ }
  public void onStatusChanged(String provider, int status,
  Bundle extras){ }
 };
```

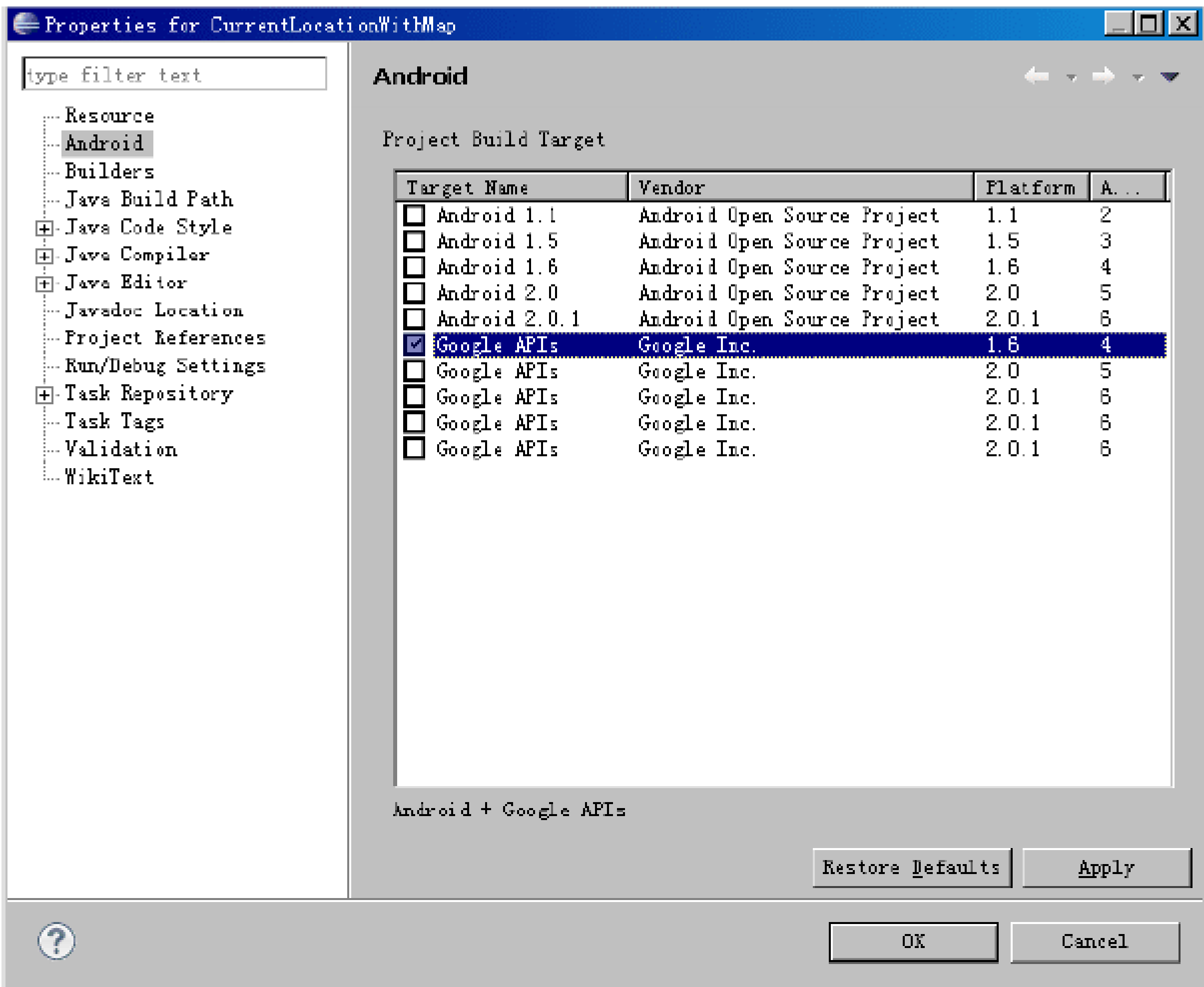
因为模拟器上没有 GPS 设备，所以需要在 Eclipse 的 DDMS 工具中提供模拟的 GPS 数据。即选择 DDMS | Emulator Control 命令，在弹出的对话框中找到 Location Control 选项，在此输入坐标，完成后单击 Send 按钮，如图 13-1 所示。



图 13-1　设置坐标

因为用到了 Google API，所以要在项目中引入 Google API，邮件单击项目选择 Properties，在弹出的对话框中选择 Google APIs 版本，如图 13-2 所示。

这样模拟器运行后，会显示当前的坐标，如图 13-3 所示。



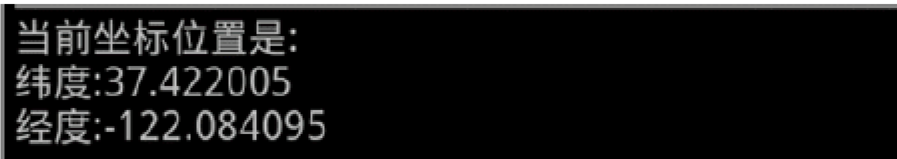


图 13-2　引用 Google API

图 13-3　执行效果

13.2　使用谷歌地图

实例 147	在手机中使用谷歌地图
源码路径	光盘:\daima\147
视频路径	光盘:\视频\147
实例必备	147.随时更新位置信息.pdf ① 库 Maps 中的类 ② 使用 LocationManager 监听位置

13.2.1　实例说明

在本实例中，使用 Map API 密钥在手机屏幕中显示谷歌地图。Google 地图给人们的生活带来了极大的方便，例如，可以通过 Google 地图查找商户信息、查看地图和获取行车路线等。Android 平台也提供了一个 map 包（com.google.android.maps），通过其中的 MapView 能够方便地利用 Google 地图的

资源进行编程。

在使用谷歌地图之前需要预先进行如下设置。

1．添加 maps.jar 到项目

在 Android SDK 中，以 JAR 库的形式提供了和 MAP 有关的 API，此 JAR 库位于 android-sdk-windows\add-ons\google_apis-4 目录下。要把 maps.jar 添加到项目中，可以在项目属性中的 Android 栏中指定使用包含 Google API 的 Target 作为项目的构建目标，如图 13-4 所示。

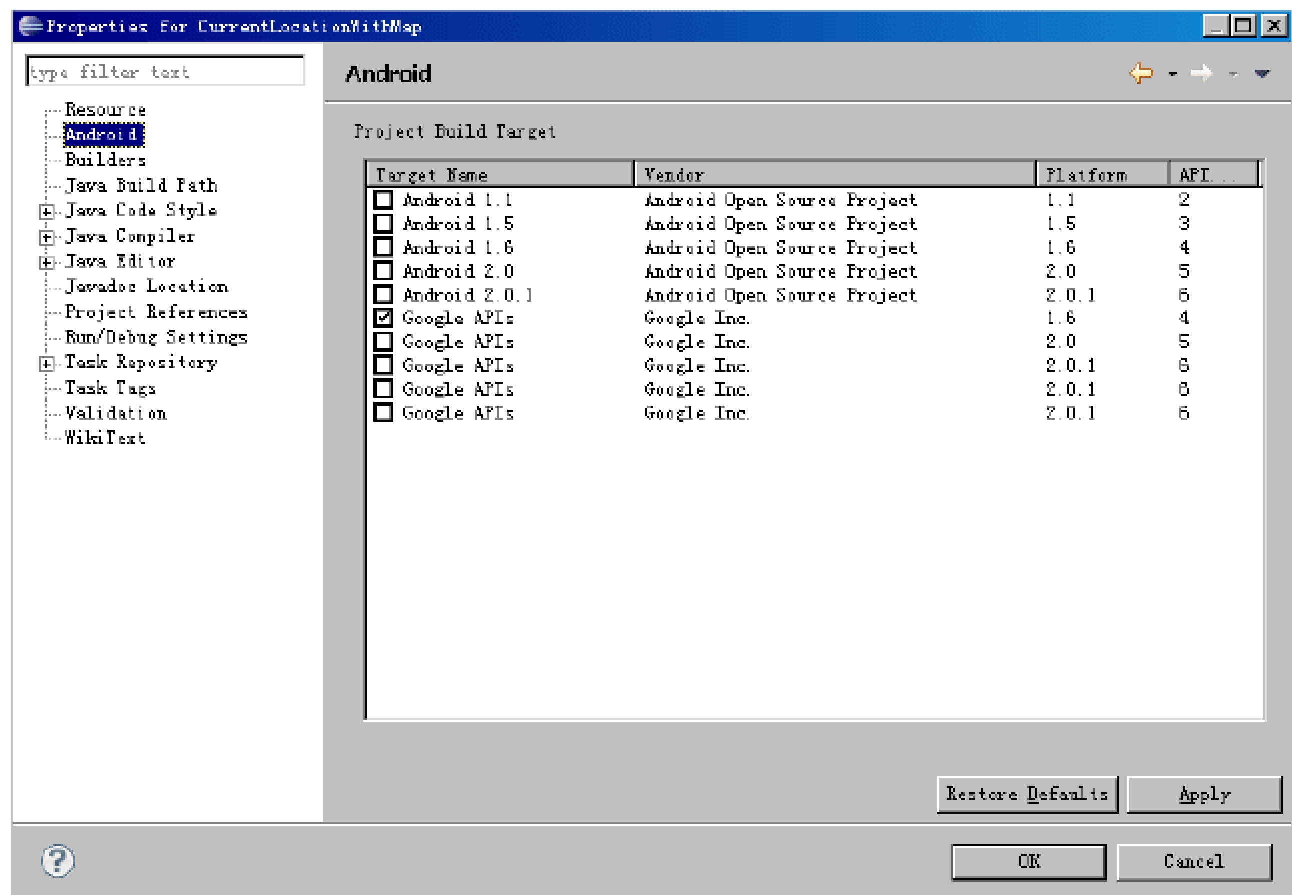


图 13-4　在项目中包含 Google API

2．将地图嵌入到应用

通过使用 MapActivity 和 MapView 控件，可以轻松地将地图嵌入到应用程序中。在此步骤中，需要将 Google API 添加到构建路径中。方法是在图 13-4 所示界面中选择 Java Build Path，然后在 Target 中选中 Google APIs 复选框，设置项目中包含 Google APIs，如图 13-5 所示。

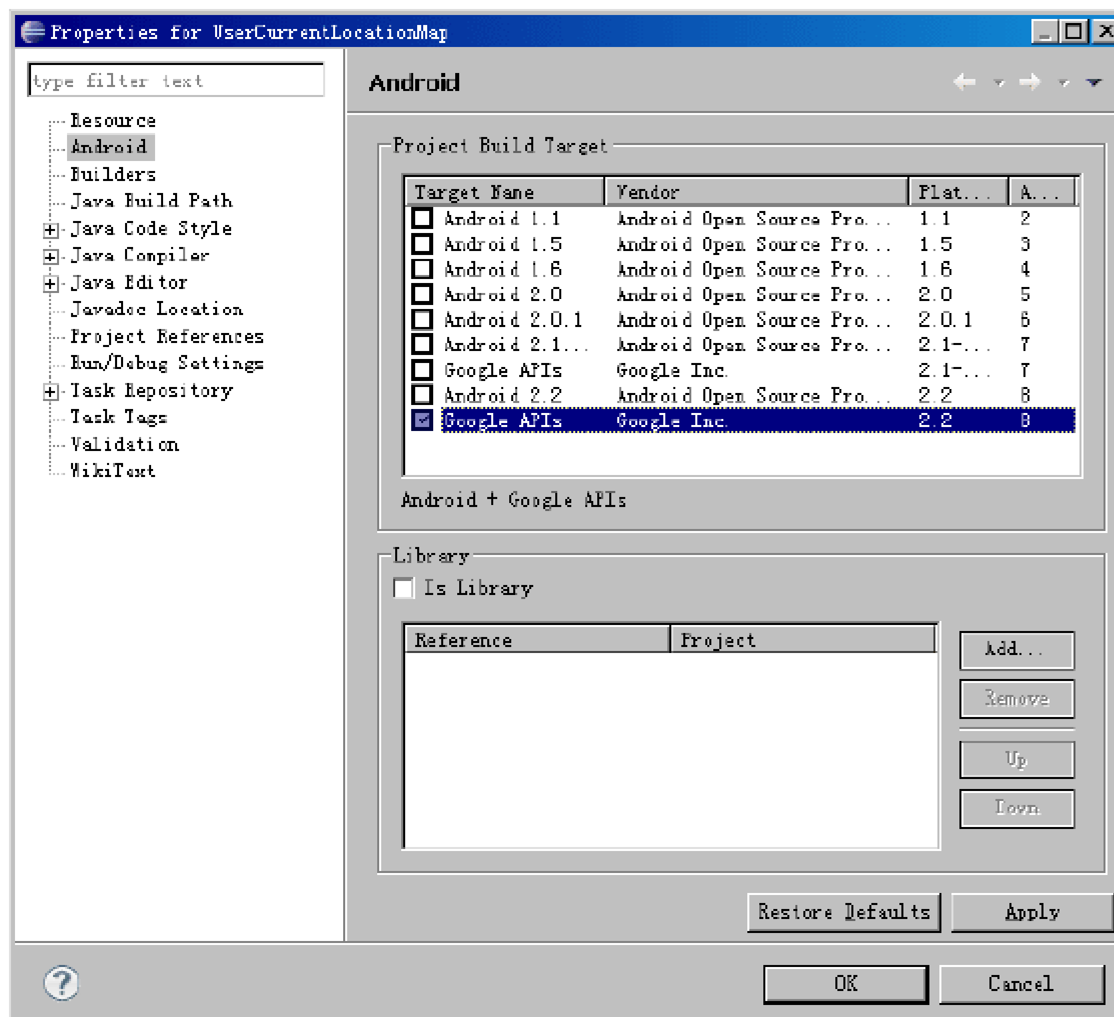


图 13-5　将 Google API 添加到构建路径

3．获取 Map API 密钥

在使用 MapView 之前，必须先申请一个 Android Map API Key，具体步骤如下所示。

（1）找到 debug.keystore 文件，通常位于如下目录中。

```
C:\Documents and Settings\当前用户\Local Settings\Application Data\Android
```

（2）运行 cmd.exe，执行如下命令获取 MD5 指纹。

```
>keytool -list -alias androiddebugkey -keystore "debug.keystore 的路径" -storepass android -keypass android
```

例如笔者在个人机器中输入如下命令。

```
keytool -list -alias androiddebugkey -keystore "C:\Documents and Settings\Administrator\.android\debug.keystore"
-storepass android -keypass android
```

此时系统会提示输入 keystore 密码，输入 Android 后系统会输出申请到的 MD5 认证指纹，如图 13-6 所示。

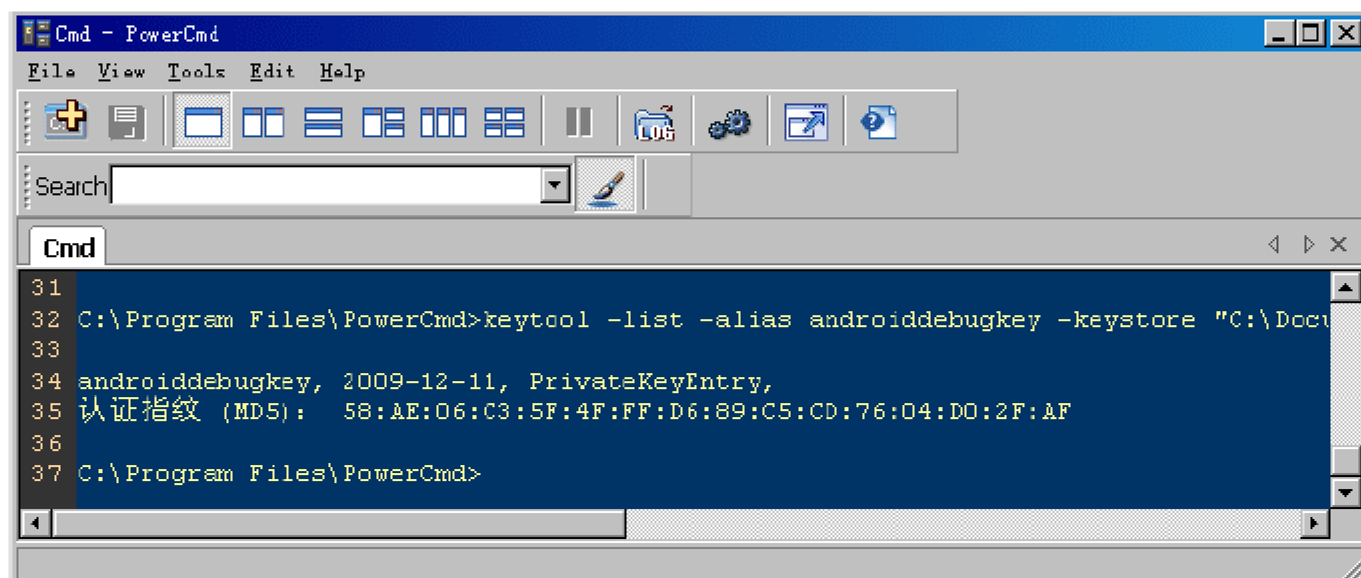


图 13-6　获取的认证指纹

注意： 因为在 CMD 中不能直接复制、粘贴 CMD 命令，这样很影响编程效率，所以笔者使用了第三方软件 PowerCmd 来代替机器中自带的 CMD 工具。

（3）开始申请 Android map 的 API Key。

在浏览器中输入网址 http://code.google.com/intl/zh-CN/android/maps-api-signup.html，如图 13-7 所示。

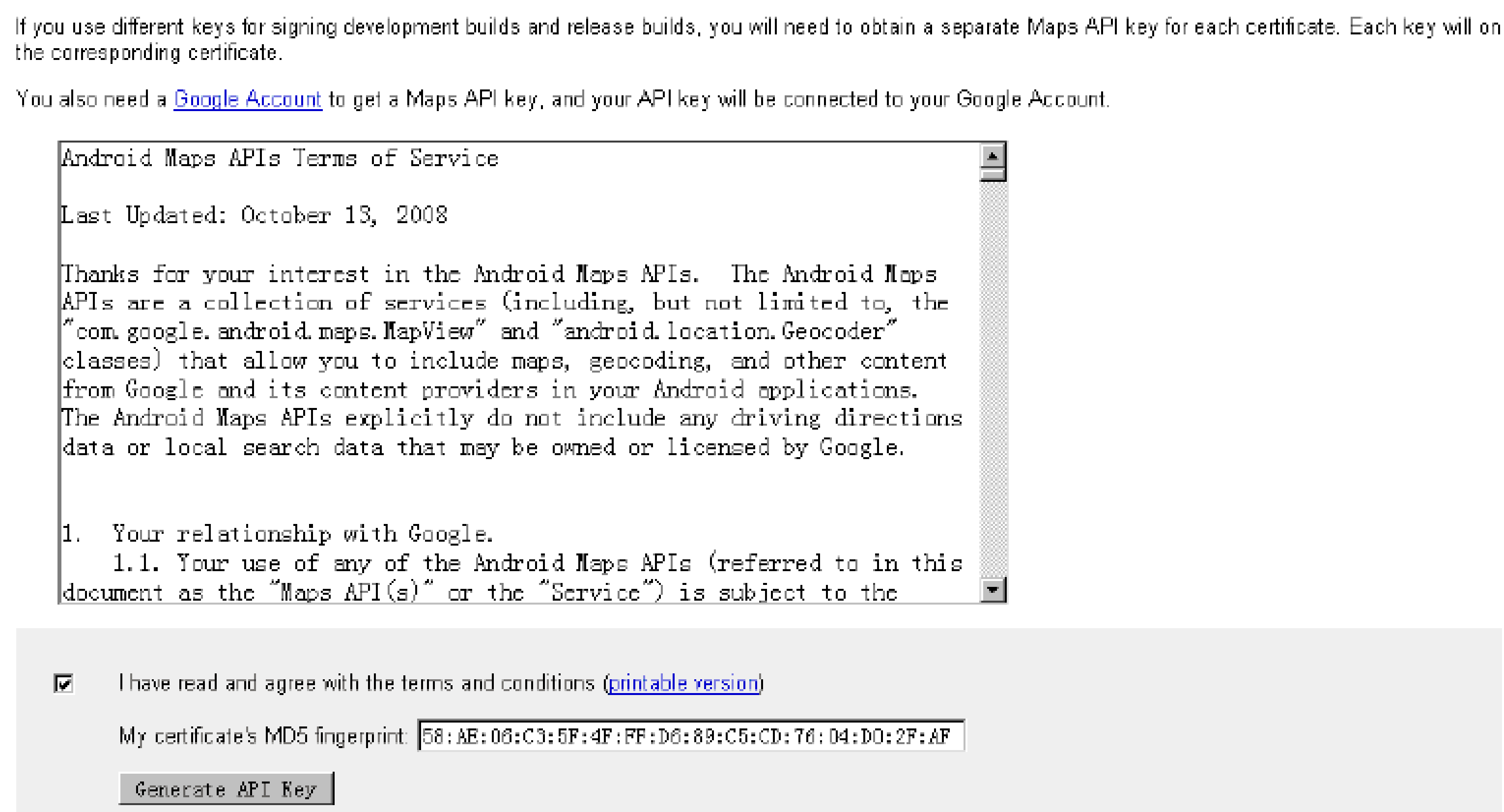


图 13-7　申请主页

在谷歌的 android map API Key 申请页面上输入图 13-6 中得到的 MD5 认证指纹，单击 Generate API Key 按钮转到图 13-8 所示的界面，下方是申请到的 API Key。



图 13-8　得到的 API Key

13.2.2　具体实现

（1）编写主布局文件 main.xml，在其中插入两个 Button，分别用于放大和缩小地图，然后通过 ToggleButton 来控制是否显示卫星地图，在最后设置申请的 apiKey，主要代码如下所示。

```
<ToggleButton
    android:id="@+id/switchMap"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="卫星视图(关)"
    android:textOn="卫星视图(开)"/>
<com.google.android.maps.MapView
     android:id="@+id/myMapView"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:clickable="true"
     android:apiKey="0by7ffx8jX0A_LWXeKCMTWAh8CqHAlqvzetFqjQ"
     />
```

（2）在文件 AndroidManifest.xml 中声明 INTERNET 和 ACCESS_FINE_LOCATION 权限，主要代码如下所示。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

（3）编写文件 example.java，其实例说明流程如下所示。

① 将 MapView 绘制到屏幕上，因为 MapView 只能继承自 MapActivity 的活动中，所以必须用方法 onCreate()将 MapView 绘制到屏幕上，并同时覆盖方法 isRouteDisplayed()，表示是否需要在地图上绘制导航线路，具体代码如下所示。

```
public class example extends MapActivity {
    MapView map;
    MapController ctrlMap;
    Button inBtn;
    Button outBtn;
    ToggleButton switchMap;
    @Override
     protected boolean isRouteDisplayed() {
          return false;
     }
```

② 引入主布局文件 main.xml，调用 getOverlays()方法获取其 Overlay 链表，并将构建好的

MyLocationOverlay 对象添加到链表中。其中，MyLocationOverlay 对象调用的 enableMyLocation()方法表示尝试通过位置服务来获取当前的位置，具体代码如下所示。

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        map = (MapView)findViewById(R.id.myMapView);
        List<Overlay> overlays = map.getOverlays();
        MyLocationOverlay myLocation = new MyLocationOverlay(this,map);
        myLocation.enableMyLocation();
        overlays.add(myLocation);
```

③ 为“放大”和“缩小”这两个按钮设置单击响应程序，首先通过方法 getController()获取 MapView 的 MapController 对象，然后在“放大”和“缩小”两个按钮单击事件监听器的回放方法中，根据按钮的不同实现对 MapView 的缩放，具体代码如下所示。

```
        ctrlMap = map.getController();
        inBtn = (Button)findViewById(R.id.in);
        outBtn = (Button)findViewById(R.id.out);
        OnClickListener listener = new OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (v.getId()) {
                case R.id.in:                                    /*如果是缩小*/
                    ctrlMap.zoomIn();
                    break;
                case R.id.out:                                   /*如果是放大*/
                    ctrlMap.zoomOut();
                    break;
                default:
                    break;
                }
            }
        };
        inBtn.setOnClickListener(listener);
        outBtn.setOnClickListener(listener);

        //=======================================
```

④ 通过方法 onCheckedChanged()获取是否选择了 switchMap 卫星视图，如果选择了 switchMap 卫星视图，则显示卫星地图。首先通过方法 findViewById()获取对应 id 的 ToggleButton 对象的引用，然后调用 setOnCheckedChange Listener()方法，设置对事件监听器选中的事件进行处理。根据 ToggleButton 是否被选中，进而通过 setSatellite()方法启用或禁用卫星视图功能，具体代码如下所示。

```
       switchMap = (ToggleButton)findViewById(R.id.switchMap);
        switchMap.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton cBtn, boolean isChecked) {
                if (isChecked == true) {
                    map.setSatellite(true);
                } else {
                    map.setSatellite(false);
                }
            }
        });
```

⑤ 使用 LocationManager 获取当前的位置，然后通过方法 getBestProvider()获取查询条件，最后设置更新位置信息的最小间隔为 2 秒，位移变化在 10 米以上，具体代码如下所示。

```
            LocationManager locationManager;
            String context = Context.LOCATION_SERVICE;
            locationManager = (LocationManager)getSystemService(context);
            Criteria criteria = new Criteria();
            criteria.setAccuracy(Criteria.ACCURACY_FINE);
            criteria.setAltitudeRequired(false);
            criteria.setBearingRequired(false);
            criteria.setCostAllowed(true);
            criteria.setPowerRequirement(Criteria.POWER_LOW);
            String provider = locationManager.getBestProvider(criteria, true);
            Location location = locationManager.getLastKnownLocation(provider);
            updateWithNewLocation(location);
            locationManager.requestLocationUpdates(provider, 2000, 10,
                    locationListener);
```

⑥ 定义方法 updateWithNewLocation(Location location)显示地理信息和地图信息，具体代码如下所示。

```
      private void updateWithNewLocation(Location location) {
       String latLongString;
       TextView myLocationText;
       myLocationText = (TextView)findViewById(R.id.myLocationText);
       if (location != null) {
             double lat = location.getLatitude();
             double lng = location.getLongitude();
             latLongString = "纬度:" + lat + "\n 经度:" + lng;
             ctrlMap.animateTo(new GeoPoint((int)(lat*1E6),(int)(lng*1E6)));
       } else {
             latLongString = "无法获取地理信息";
       }
       myLocationText.setText("您当前的位置是:\n" +
       latLongString);
      }
}
```

此时在图 13-9 中选定一个经度和纬度位置后可以显示此位置的定位信息，并且定位信息分别以文字和地图形式显示出来，如图 13-10 所示。

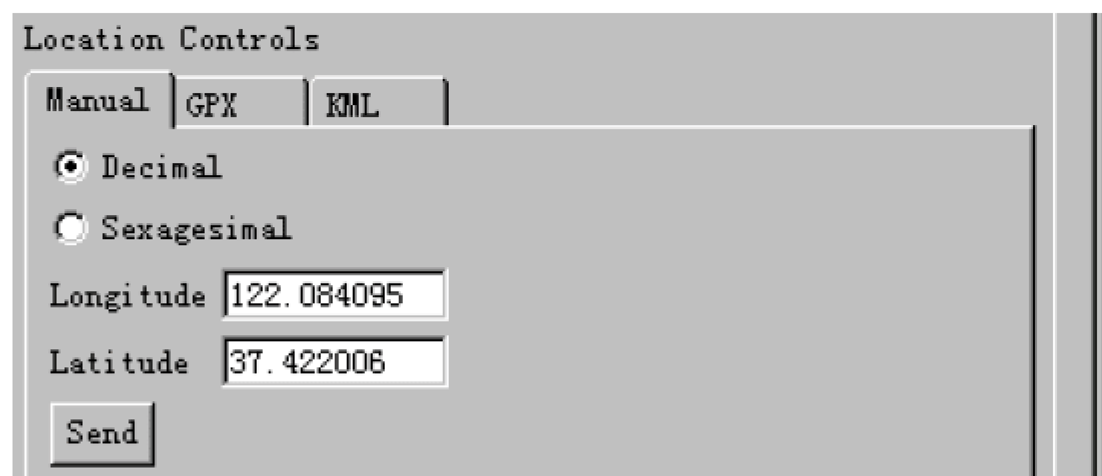


图 13-9　指定位置



图 13-10　显示对应信息

单击“放大”和“缩小”按钮后可以控制地图的大小，如图 13-11 所示。打开卫星视图后可以显示此位置范围对应的卫星地图，如图 13-12 所示。

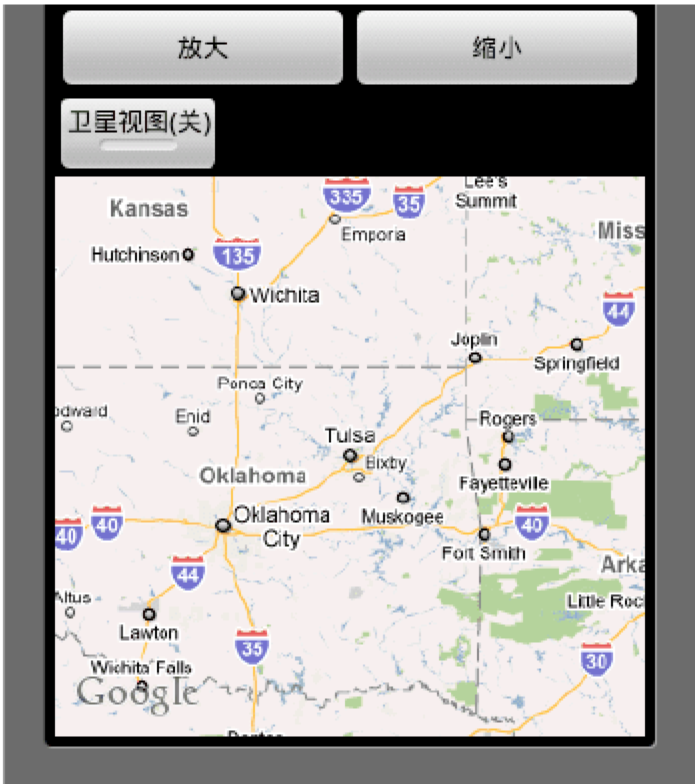


图 13-11　放大后效果

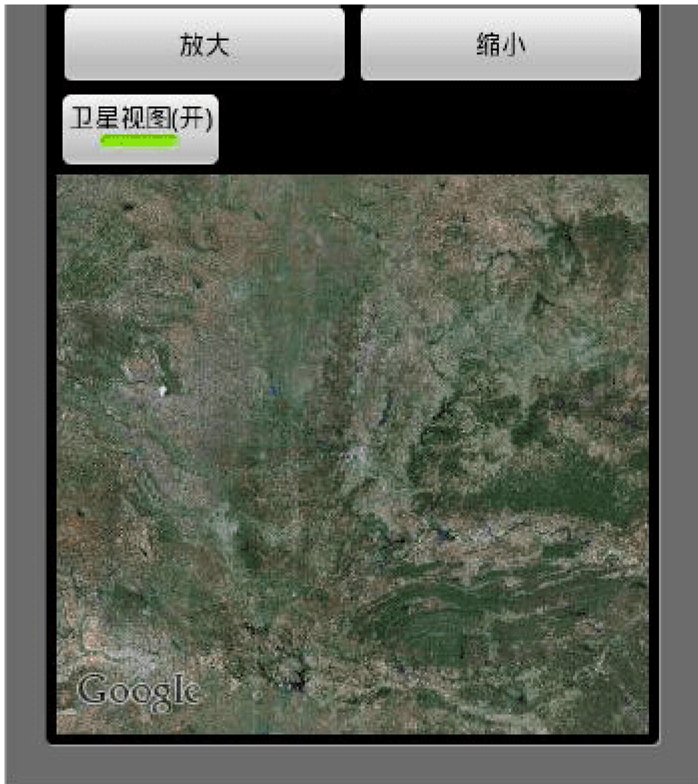


图 13-12　卫星地图

13.3　输入一个坐标后在地图中实现定位

实例 148	输入一个坐标后在地图中实现定位
源码路径	光盘:\daima\148
视频路径	光盘:\视频\148
实例必备	148.在 Android 设备中使用地图.pdf ① 添加 Google Map 密钥 ② 使用 Map API 密钥

13.3.1　实例说明

在本实例中，通过 Google MapView 和 GeoPoint 实现地图经纬应用的流程。当在表单中输入一个经度和纬度值后，单击“查询”按钮会在地图中显示此位置。

13.3.2　具体实现

编写文件 example.java，在 EditText 文本框中输入坐标的经度和纬度，将坐标转换为 GeoPoint 对象后，再利用 MapController 的 animateTo()方法将地图的中心点移到 GeoPoint 坐标上。文件 example.java 的主要代码如下所示。

```
/*Map 启动时的默认坐标*/
private double dLat=120.391177;
private double dLng=39.9067452;
@Override
protected void onCreate(Bundle icicle)
{
```

```
super.onCreate(icicle);
setContentView(R.layout.main);
/*创建 MapView 对象*/
mMapView01 = (MapView)findViewById(R.id.myMapView1);
mMapController01 = mMapView01.getController();
/*设置 MapView 的显示选项（卫星、街道）*/
mMapView01.setSatellite(false);
mMapView01.setStreetView(true);
/*默认放大的层级*/
intZoomLevel = 17;
mMapController01.setZoom(intZoomLevel);
/*设置 Map 的中点为默认经纬度*/
refreshMapView();

mEditText01 = (EditText)findViewById(R.id.myEdit1);
mEditText02 = (EditText)findViewById(R.id.myEdit2);

/*送出查询的 Button*/
mButton01 = (Button)findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    /*经纬度空白检查*/
    if(mEditText01.getText().toString().equals("")||
       mEditText02.getText().toString().equals(""))
    {
      showDialog("经度或纬度填写不正确!");
    }
    else
    {
      /*取得输入的经纬度*/
      dLng=Double.parseDouble(mEditText01.getText().toString());
      dLat=Double.parseDouble(mEditText02.getText().toString());
      /*依输入的经纬度重整 Map*/
      refreshMapView();
    }
  }
});

/*放大 Map 的 Button*/
mButton02 = (Button)findViewById(R.id.myButton2);
mButton02.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    intZoomLevel++;
    if(intZoomLevel>mMapView01.getMaxZoomLevel())
    {
```

```
                intZoomLevel = mMapView01.getMaxZoomLevel();
            }
            mMapController01.setZoom(intZoomLevel);
        }
    });

    /*缩小 Map 的 Button*/
    mButton03 = (Button)findViewById(R.id.myButton3);
    mButton03.setOnClickListener(new Button.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            intZoomLevel--;
            if(intZoomLevel<1)
            {
                intZoomLevel = 1;
            }
            mMapController01.setZoom(intZoomLevel);
        }
    });
}

/*重整 Map 的 method*/
public void refreshMapView()
{
    GeoPoint p = new GeoPoint((int)(dLat* 1E6), (int)(dLng* 1E6));
    mMapView01.displayZoomControls(true);
    /*将 Map 的中点移至 GeoPoint*/
    mMapController01.animateTo(p);
    mMapController01.setZoom(intZoomLevel);
}

@Override
protected boolean isRouteDisplayed()
{
    return false;
}

/*显示 Dialog 的 method*/
private void showDialog(String mess){
    new AlertDialog.Builder(example189.this).setTitle("Message")
    .setMessage(mess)
    .setNegativeButton("确定", new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int which)
        {
        }
    })
    .show();
}
}
```

执行后的效果如图 13-13 所示。



图 13-13　执行效果

13.4　实现地址查询功能

实例 149	在手机中实现地址查询
源码路径	光盘:\daima\149
视频路径	光盘:\视频\149
实例必备	149.接近警报.pdf ① 类 Geocoder 基础 ② Geocoder 的公共构造器和公共方法

13.4.1　实例说明

在 Google 中提供了一个 Geocoder 服务，在非商用情况下，使用 Geocoder 可以反查 Address 地址对象服务。在本实例中，通过使用 Geocoder 实现地址反查功能。

13.4.2　具体实现

编写文件 example.java，在此文件中通过地址来获取 GeoPoint()方法，在自定义函数 getGeoByAddress 中传入唯一的值字符串的方式传入地址，使用方法 Geocoder.getFromLocationName()来获取从 Google 服务器中找到的结果。文件 example.java 的主要代码如下所示。

```
protected void onCreate(Bundle icicle)
{
  super.onCreate(icicle);
  setContentView(R.layout.main);

  mEditText01 = (EditText)findViewById(R.id.myEditText1);
  mEditText01.setText
```

```
(
   getResources().getText(R.string.str_default_address).toString()
);

/*创建 MapView 对象*/
mMapView01 = (MapView)findViewById(R.id.myMapView1);
mMapController01 = mMapView01.getController();
//设置 MapView 的显示选项（卫星、街道）
mMapView01.setSatellite(true);
mMapView01.setStreetView(true);
mButton01 = (Button)findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
   @Override
   public void onClick(View v)
   {
      if(mEditText01.getText().toString()!="")
      {
         refreshMapViewByGeoPoint
         (
            getGeoByAddress
            (
               mEditText01.getText().toString()
            ),mMapView01,intZoomLevel,true
         );
      }
   }
});

/*放大*/
mButton02 = (Button)findViewById(R.id.myButton2);
mButton02.setOnClickListener(new Button.OnClickListener()
{
   @Override
   public void onClick(View v)
   {
      // TODO Auto-generated method stub
      intZoomLevel++;
      if(intZoomLevel>mMapView01.getMaxZoomLevel())
      {
         intZoomLevel = mMapView01.getMaxZoomLevel();
      }
      mMapController01.setZoom(intZoomLevel);
   }
});

/*缩小*/
mButton03 = (Button)findViewById(R.id.myButton3);
mButton03.setOnClickListener(new Button.OnClickListener()
{
   @Override
   public void onClick(View v)
   {
```

```
            // TODO Auto-generated method stub
            intZoomLevel--;
            if(intZoomLevel<1)
            {
              intZoomLevel = 1;
            }
            mMapController01.setZoom(intZoomLevel);
          }
        });

        /*初次查询地点*/
        refreshMapViewByGeoPoint
        (
          getGeoByAddress
          (
              getResources().getText(R.string.str_default_address).toString()
          ),mMapView01,intZoomLevel,true
        );
      }
      private GeoPoint getGeoByAddress(String strSearchAddress)
      {
        GeoPoint gp = null;
        try
        {
          if(strSearchAddress!="")
          {
            Geocoder mGeocoder01 = new Geocoder(example190.this, Locale.getDefault());
            List<Address> lstAddress = mGeocoder01.getFromLocationName(strSearchAddress, 1);
            if (!lstAddress.isEmpty())
            {
              //Address[addressLines=[0:"U.S PIZZA",1:"15th Main Rd, Phase II, J P Nagar",2:"Bengaluru,
Karnataka",3:"India"],feature=U.S PIZZA,admin=Karnataka,sub-admin=Bengaluru,locality=Bengaluru, thoroughfare=
15th Main Rd,postalCode=null,countryCode=IN,countryName=India, hasLatitude=true,latitude=18.508933, hasLongitude=
true,longitude=73.8042,phone=null,url=null,extras=null]
              /*
              for (int i = 0; i < lstAddress.size(); ++i)
              {
                Address adsLocation = lstAddress.get(i);
                Log.i(TAG, "Address found = " + adsLocation.toString());
              }
              Address adsLocation = lstAddress.get(0);
              double geoLatitude = adsLocation.getLatitude()*1E6;
              double geoLongitude = adsLocation.getLongitude()*1E6;
              gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
            }
            else
            {
              Log.i(TAG, "Address GeoPoint NOT Found.");
            }
          }
        }
        catch (Exception e)
        {
```

```
            e.printStackTrace();
        }
        return gp;
    }
    public static void refreshMapViewByGeoPoint(GeoPoint gp, MapView mv, int zoomLevel, boolean bIfSatellite)
    {
        try
        {
            mv.displayZoomControls(true);
            /*取得 MapView 的 MapController*/
            MapController mc = mv.getController();
            /*移至该地理坐标地址*/
            mc.animateTo(gp);

            /*放大地图层级*/
            mc.setZoom(zoomLevel);
            /*设置 MapView 的显示选项（卫星、街道）*/
            if(bIfSatellite)
            {
                mv.setSatellite(true);
                mv.setStreetView(true);
            }
            else
            {
                mv.setSatellite(false);
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    @Override
    protected boolean isRouteDisplayed()
    {
        // TODO Auto-generated method stub
        return false;
    }
}
```

执行后能够实现地址反查处理，如图 13-14 所示。



图 13-14　执行效果

13.5　实现路径导航

实例 150	在手机屏幕中实现路径导航
源码路径	光盘:\daima\150
视频路径	光盘:\视频\150
实例必备	150.起点和终点的设置.pdf

13.5.1　实例说明

在 Android SDK 中，可以使用手机内置地图程序传递导航坐标的方式来规划路径。在本实例中，通过 Directions Route 实现了路径导航功能。在实例说明上，先调用 getLocationProvider()获取当前 Location，以取得当前所在位置的地理坐标，并通过提供的 EditText Widget 让用户输入将要前往的地址，通过地址取得目的地的地理坐标。通过两个 GeoPoint 对象，并通过 Intent 方式调用内置的地图程序。

13.5.2　具体实现

编写文件 example.java，其实例说明流程如下所示。

（1）创建 LocationManager 对象以获取系统 LOCATION 本地服务，然后设置使用 MapView 控件显示选项（卫星、街道），主要代码如下所示。

```
protected void onCreate(Bundle icicle)
{
  // TODO Auto-generated method stub
  super.onCreate(icicle);
  setContentView(R.layout.main);

  mTextView01 = (TextView)findViewById(R.id.myTextView1);

  mEditText01 = (EditText)findViewById(R.id.myEditText1);
  mEditText01.setText
  (
    getResources().getText
    (R.string.str_default_address).toString()
  );

  /*创建 MapView 对象*/
  mMapView01 = (MapView)findViewById(R.id.myMapView1);
  mMapController01 = mMapView01.getController();

  //设置 MapView 的显示选项（卫星、街道）
  mMapView01.setSatellite(true);
  mMapView01.setStreetView(true);
```

```
//放大的层级
intZoomLevel = 15;
mMapController01.setZoom(intZoomLevel);

/*创建 LocationManager 对象取得系统 LOCATION 服务*/
mLocationManager01 =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);

/*
 * 自定义函数，访问 Location Provider,
 * 并将之存储在 strLocationProvider 中
 */
getLocationProvider();

/*传入 Location 对象，显示于 MapView*/
fromGeoPoint = getGeoByLocation(mLocation01);
refreshMapViewByGeoPoint(fromGeoPoint,
                   mMapView01, intZoomLevel);

/*创建 LocationManager 对象，监听
 * Location 更改时事件，更新 MapView*/
mLocationManager01.requestLocationUpdates
(strLocationProvider, 2000, 10, mLocationListener01);
```

（2）定义单击 mButton01 按钮的处理事件，先获取用户要前往地址的 GeoPoint 对象，传入路径规划所需要的地标地址，实例说明代码如下所示。

```
mButton01 = (Button)findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    // TODO Auto-generated method stub
    if(mEditText01.getText().toString()!="")
    {
      /*取得 User 要前往地址的 GeoPoint 对象*/
      toGeoPoint =
      getGeoByAddress(mEditText01.getText().toString());

      /*路径规划 Intent*/
      Intent intent = new Intent();
      intent.setAction(android.content.Intent.ACTION_VIEW);

      /*传入路径规划所需要的地标地址*/
      intent.setData
      (
        Uri.parse("http://maps.google.com/maps?f=d&saddr="+
        GeoPointToString(fromGeoPoint)+
        "&daddr="+GeoPointToString(toGeoPoint)+
        "&hl=cn" +
        "")
```

```
            );
            startActivity(intent);
          }
        }
      });
```

（3）定义单击 mButton02 按钮的处理事件，单击后实现地图放大处理，具体代码如下所示。

```
      /*放大地图*/
      mButton02 = (Button)findViewById(R.id.myButton2);
      mButton02.setOnClickListener(new Button.OnClickListener()
      {
        @Override
        public void onClick(View v)
        {
          // TODO Auto-generated method stub
          intZoomLevel++;
          if(intZoomLevel>mMapView01.getMaxZoomLevel())
          {
            intZoomLevel = mMapView01.getMaxZoomLevel();
          }
          mMapController01.setZoom(intZoomLevel);
        }
      });
```

（4）定义单击 mButton03 按钮的处理事件，单击实现地图缩小处理，具体代码如下所示。

```
      /*缩小地图*/
      mButton03 = (Button)findViewById(R.id.myButton3);
      mButton03.setOnClickListener(new Button.OnClickListener()
      {
        @Override
        public void onClick(View v)
        {
          // TODO Auto-generated method stub
          intZoomLevel--;
          if(intZoomLevel<1)
          {
            intZoomLevel = 1;
          }
          mMapController01.setZoom(intZoomLevel);
        }
      });
    }
```

（5）捕捉当手机 GPS 坐标更新时的事件，在手机收到位置更改时，将 location 传入 getMyLocation 对象，具体代码如下所示。

```
    /*捕捉当手机 GPS 坐标更新时的事件*/
    public final LocationListener mLocationListener01 =
    new LocationListener()
    {
      @Override
      public void onLocationChanged(Location location)
      {
        /*当手机收到位置更改时，将 location 传入 getMyLocation*/
```

```
            mLocation01 = location;
            fromGeoPoint = getGeoByLocation(location);
            refreshMapViewByGeoPoint(fromGeoPoint,
                  mMapView01, intZoomLevel);
        }
    };
```

（6）定义方法 getGeoByLocation()，设置当传入 Location 对象时取回其 GeoPoint 对象，具体代码如下所示。

```
/*传入 Location 对象，取回其 GeoPoint 对象*/
private GeoPoint getGeoByLocation(Location location)
{
  GeoPoint gp = null;
  try
  {
    /*如果 Location 存在*/
    if (location != null)
    {
      double geoLatitude = location.getLatitude()*1E6;
      double geoLongitude = location.getLongitude()*1E6;
      gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
    }
  }
  catch(Exception e)
  {
    e.printStackTrace();
  }
  return gp;
}
```

（7）定义方法 getGeoByAddress()，设置当输入地址时获取其 GeoPoint 对象，具体代码如下所示。

```
/*输入地址，获取其 GeoPoint 对象*/
private GeoPoint getGeoByAddress(String strSearchAddress)
{
  GeoPoint gp = null;
  try
  {
    if(strSearchAddress!="")
    {
      Geocoder mGeocoder01 = new Geocoder
      (example191.this, Locale.getDefault());

      List<Address> lstAddress = mGeocoder01.getFromLocationName
                         (strSearchAddress, 1);
      if (!lstAddress.isEmpty())
      {
        Address adsLocation = lstAddress.get(0);
        double geoLatitude = adsLocation.getLatitude()*1E6;
        double geoLongitude = adsLocation.getLongitude()*1E6;
        gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
      }
```

```
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return gp;
    }
```

（8）定义方法 refreshMapViewByGeoPoint()和 refreshMapViewByCode()，分别用于传入 geoPoint 更新 MapView 中的谷歌地图和传入经纬度更新 MapView 中的谷歌地图，具体代码如下所示。

```
    /*传入 geoPoint 更新 MapView 中的 Google Map*/
    public static void refreshMapViewByGeoPoint
    (GeoPoint gp, MapView mapview, int zoomLevel)
    {
        try
        {
            mapview.displayZoomControls(true);
            MapController myMC = mapview.getController();
            myMC.animateTo(gp);
            myMC.setZoom(zoomLevel);
            mapview.setSatellite(false);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    /*传入经纬度更新 MapView 中的 Google Map*/
    public static void refreshMapViewByCode
    (double latitude, double longitude,
            MapView mapview, int zoomLevel)
    {
        try
        {
            GeoPoint p = new GeoPoint((int) latitude, (int) longitude);
            mapview.displayZoomControls(true);
            MapController myMC = mapview.getController();
            myMC.animateTo(p);
            myMC.setZoom(zoomLevel);
            mapview.setSatellite(false);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
```

（9）定义方法 GeoPointToString()，将 GeoPoint 中的经纬度以 String,String 的格式返回，具体代码

如下所示。

```
/*将 GeoPoint 里的经纬度以 String,String 返回*/
private String GeoPointToString(GeoPoint gp)
{
  String strReturn="";
  try
  {
    /*当 Location 存在*/
    if (gp != null)
    {
      double geoLatitude = (int)gp.getLatitudeE6()/1E6;
      double geoLongitude = (int)gp.getLongitudeE6()/1E6;
      strReturn = String.valueOf(geoLatitude)+","+
        String.valueOf(geoLongitude);
    }
  }
  catch(Exception e)
  {
    e.printStackTrace();
  }
  return strReturn;
}
```

执行后的效果如图 13-15 所示；单击“开始规划路径”按钮后弹出选择对话框，如图 13-16 所示。在此选择 Maps 后弹出规划界面，如图 13-17 所示；在图 13-17 所示界面中，在第一个文本框中设置出发位置，如 beijing，在第二个文本框中设置目的地位置，如 tianjin，单击按钮，如图 13-18 所示；单击 Go 按钮，系统将实现从北京到天津的线路规划，产生线路规划图，最终的执行界面如图 13-19 所示。



图 13-15　执行效果

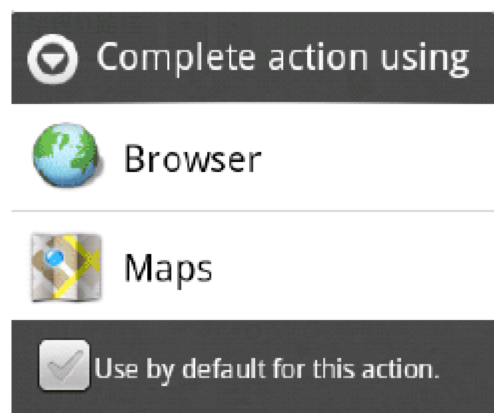


图 13-16　选择对话框

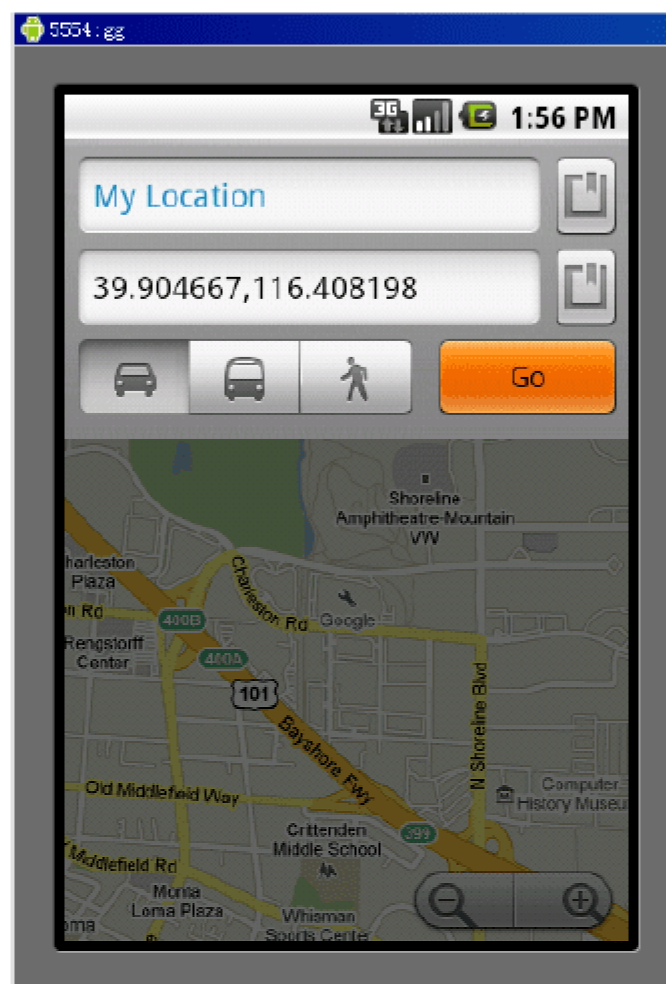


图 13-17　规划界面

单击图 13-19 中的 Show on map 按钮，将会在地图中显示行走线路，如图 13-20 所示。

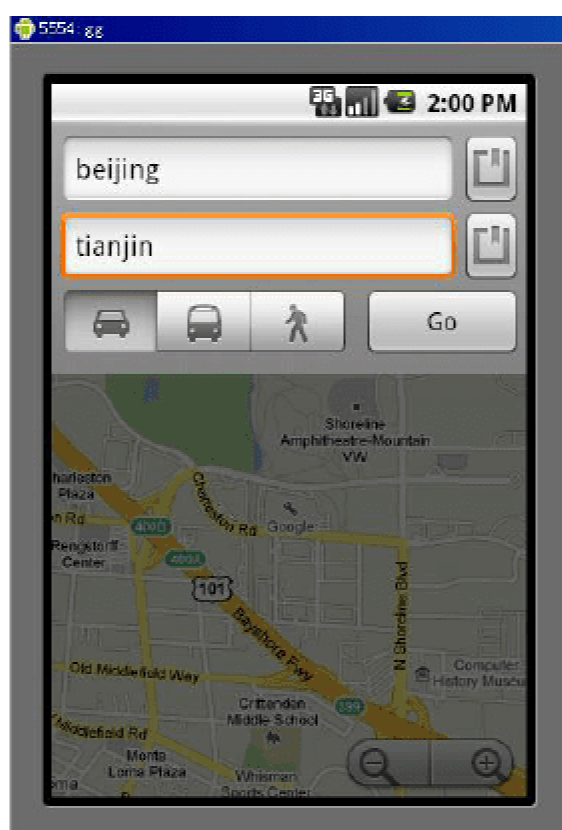


图 13-18　设置出发地和目的地



图 13-19　生成的线路规划



图 13-20　地图中的线路规划

在此需要注意，出发地和目的地不能属于两个不同的国家，否则将会产生错误提示。

13.6　移动手机时自动实现位置更新

实例 151	移动手机时自动实现位置更新
源码路径	光盘:\daima\151
视频路径	光盘:\视频\151
实例必备	151.判断 GPS 模块是否存在或开启.pdf

13.6.1　实例说明

在现实应用中，GPS 的使用越来越广泛。但是每一个位置和路况都不是固定不变的，这就要求系统能够根据各种变化而变化，不能误导用户。在 Android SDK 中，支持手机 GPS 定位事件处理。在

Android 手机中内置了 Google Map，但是不能随着手机的移动而更新。在本实例中，将插入一个 TextView 和 MapView，当手机移动时，会触发内置的 GPS 定位坐标的改变事件。只要程序发现地理位置的变化，便实时更新 MapView 中的 Google Map，并反查地理坐标系统的信息，从而实现了在 Android 中 GPS 实时更新的功能。

13.6.2　具体实现

编写文件 example.java，其实例说明流程如下所示。

（1）创建 LocationManager 对象 mLocationManager01 来获取系统 LOCATION 服务，具体代码如下所示。

```
mTextView01 = (TextView)findViewById(R.id.myTextView1);
/*创建 MapView 对象*/
mMapView01 = (MapView)findViewById(R.id.myMapView1);

/*创建 LocationManager 对象取得系统 LOCATION 服务*/
mLocationManager01 =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
```

（2）通过方法 getLocationProvider()获取当前 Location 位置，创建 LocationManager 对象用于监听 Location 更改时事件，并更新 MapView 控件中的地图，具体代码如下所示。

```
  /*第一次运行向 Location Provider 获取 Location*/
  mLocation01 = getLocationProvider(mLocationManager01);

  if(mLocation01!=null)
  {
    processLocationUpdated(mLocation01);
  }
  else
  {
    mTextView01.setText
    (
      getResources().getText(R.string.str_err_location).toString()
    );
  }
  /*创建 LocationManager 对象，监听 Location 更改时事件，更新 MapView*/
  mLocationManager01.requestLocationUpdates
  (strLocationProvider, 2000, 10, mLocationListener01);
}
```

（3）通过方法 LocationListener()监听定位信息，当手机收到位置更改时将 location 传入并取得当前地理坐标，具体代码如下所示。

```
public final LocationListener
mLocationListener01 = new LocationListener()
{
  @Override
  public void onLocationChanged(Location location)
  {
    /*当手机收到位置更改时，将 location 传入并取得地理坐标*/
    processLocationUpdated(location);
```

```
    }
    public void onProviderDisabled(String provider)
    {
      /*当 Provider 已离开服务范围时*/
    }
    public void onProviderEnabled(String provider)
    {
      // TODO Auto-generated method stub
    }
```

（4）定义方法 getAddressbyGeoPoint(GeoPoint gp)获取定位地址的信息，先创建 Geocoder 对象并取出地理坐标经纬度，然后判断地址是否为多行，最后将获取的地址组合后放在 StringBuilder 对象中输出，具体代码如下所示。

```
  public String getAddressbyGeoPoint(GeoPoint gp)
  {
    String strReturn = "";
    try
    {
      /*当 GeoPoint 不等于 null*/
      if (gp != null)
      {
        /*创建 Geocoder 对象*/
        Geocoder gc = new Geocoder
        (example192.this, Locale.getDefault());

        /*取出地理坐标经纬度*/
        double geoLatitude = (int)gp.getLatitudeE6()/1E6;
        double geoLongitude = (int)gp.getLongitudeE6()/1E6;

        /*自经纬度取得地址（可能有多行地址）*/
        List<Address> lstAddress =
        gc.getFromLocation(geoLatitude, geoLongitude, 1);

        StringBuilder sb = new StringBuilder();

        /*判断地址是否为多行*/
        if (lstAddress.size() > 0)
        {
          Address adsLocation = lstAddress.get(0);
          for(int i=0;i<adsLocation.getMaxAddressLineIndex();i++)
          {
            sb.append(adsLocation.getAddressLine(i)).append("\n");
          }
          sb.append(adsLocation.getLocality()).append("\n");
          sb.append(adsLocation.getPostalCode()).append("\n");
          sb.append(adsLocation.getCountryName());
        }

        /*将获取的地址，组合后放在 StringBuilder 对象中用作输出*/
        strReturn = sb.toString();
      }
```

```
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    return strReturn;
  }

  public Location getLocationProvider(LocationManager lm)
  {
    Location retLocation = null;
    try
    {
      Criteria mCriteria01 = new Criteria();
      mCriteria01.setAccuracy(Criteria.ACCURACY_FINE);
      mCriteria01.setAltitudeRequired(false);
      mCriteria01.setBearingRequired(false);
      mCriteria01.setCostAllowed(true);
      mCriteria01.setPowerRequirement(Criteria.POWER_LOW);
      strLocationProvider = lm.getBestProvider(mCriteria01, true);
      retLocation = lm.getLastKnownLocation(strLocationProvider);
    }
    catch(Exception e)
    {
      mTextView01.setText(e.toString());
      e.printStackTrace();
    }
    return retLocation;
  }

  private GeoPoint getGeoByLocation(Location location)
  {
    GeoPoint gp = null;
    try
    {
      /*当 Location 存在*/
      if (location != null)
      {
        double geoLatitude = location.getLatitude()*1E6;
        double geoLongitude = location.getLongitude()*1E6;
        gp = new GeoPoint((int) geoLatitude, (int) geoLongitude);
      }
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    return gp;
  }

  public static void refreshMapViewByGeoPoint
```

```
(GeoPoint gp, MapView mv, int zoomLevel, boolean bIfSatellite)
{
  try
  {
    mv.displayZoomControls(true);
    /*获取 MapView 的 MapController*/
    MapController mc = mv.getController();
    /*移至该地理坐标地址*/
    mc.animateTo(gp);

    /*放大地图层级*/
    mc.setZoom(zoomLevel);

    /*设置 MapView 的显示选项（卫星、街道）*/
    if(bIfSatellite)
    {
      mv.setSatellite(true);
      mv.setStreetView(true);
    }
    else
    {
      mv.setSatellite(false);
    }
  }
  catch(Exception e)
  {
    e.printStackTrace();
  }
}
```

（5）定义方法 processLocationUpdated()，设置当手机获取的位置发生变化时将 location 传入 GeoPoint，并同时在 MapView 控件中更新显示地图，具体代码如下所示。

```
/*当手机收到位置更改，将 location 传入 GeoPoint 及 MapView*/
private void processLocationUpdated(Location location)
{
  /*传入 Location 对象，取得 GeoPoint 地理坐标*/
  currentGeoPoint = getGeoByLocation(location);
  /*更新 MapView 显示 Google Map*/
  refreshMapViewByGeoPoint
  (currentGeoPoint, mMapView01, intZoomLevel, true);
  mTextView01.setText
  (
    getResources().getText(R.string.str_my_location).toString()+
    "\n"+getAddressbyGeoPoint(currentGeoPoint)
  );
}
protected boolean isRouteDisplayed()
{
  // TODO Auto-generated method stub
  return false;
}
```

执行后将显示当前位置的定位信息，并能实现及时更新功能，如图 13-21 所示。

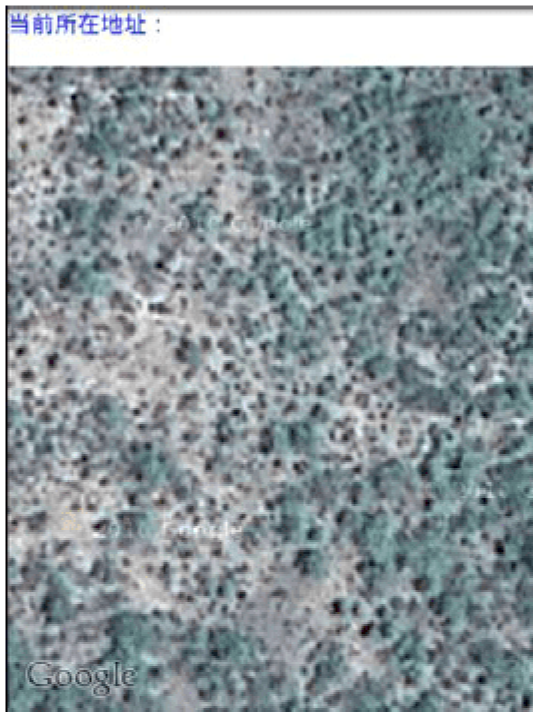


图 13-21　执行效果

13.7　模拟验证官方账号

实例 152	模拟验证官方账号
源码路径	光盘:\daima\152
视频路径	光盘:\视频\152
实例必备	152.Google Account 的组成.pdf

13.7.1　实例说明

在现实应用中，基于 Google 的大多数服务都需要通过官方进行账号验证。通过本实例的实现过程，介绍在 Android 中通过账号验证获取 Google Account Authentication Service 所发出的凭据的过程。

13.7.2　具体实现

（1）编写文件 example.java，当用户在登录 UI 中输入账号信息后获取输入的信息。在具体实现上，先通过 TextView 的 onClick()方法为起点，调用自定义方法 showLoginForm()显示登录表单。文件 example.java 的主要实现代码如下所示。

```
/*中文字的间距*/
private int intShiftPadding = 14;
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  /*创建 DisplayMetrics 对象，获取屏幕分辨率*/
  DisplayMetrics dm = new DisplayMetrics();
  getWindowManager().getDefaultDisplay().getMetrics(dm);
  mTextView01 = (TextView)findViewById(R.id.myTextView1);
```

```
    /*将文字 Label 放在屏幕右上方*/
    mTextView01.setLayoutParams
    (
      new   AbsoluteLayout.LayoutParams(intShiftPadding*mTextView01.getText().toString().length(),18,(dm.
widthPixels-(intShiftPadding*mTextView01.getText().toString().length()))-10,0)
    );
    /*用户单击 TextView 文字的事件处理——登录*/
    mTextView01.setOnClickListener(new TextView.OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
        /*显示登录对话框*/
        showLoginForm();
      }
    });
  }
  /*自定义登录对话框函数*/
  private void showLoginForm()
  {
    try
    {   /*以 LayoutInflater 取得主 Activity 的 context*/
      mInflater01 = LayoutInflater.from(example198.this);
      /*设置创建的 View 所要使用的 Layout Resource*/
      mView01 = mInflater01.inflate(R.layout.login, null);
      /*账号 EditText*/
      mEditText01=(EditText)mView01.findViewById(R.id.myEditText1);
      /*密码 EditText*/
      mEditText02=(EditText)mView01.findViewById(R.id.myEditText2);
      /*创建 AlertDialog 窗口获取用户账号密码*/
      new AlertDialog.Builder(this)
      .setView(mView01)
      .setPositiveButton("OK",
      new DialogInterface.OnClickListener()
      {
        /*覆盖 onClick()来触发获取 Token 事件与完成登录事件*/
        public void onClick(DialogInterface dialog, int whichButton)
        {
          if(processGoogleLogin(mEditText01.getText().toString(), mEditText02.getText().toString()))
          {
            Intent i = new Intent();
            /*登录后调用注销程序(example_01_02.java)*/
            i.setClass(example.this, example_01_02.class);
            startActivity(i);
            finish();
          }
        }
      }).show();
    }
    catch(Exception e)
    {
```

```
            e.printStackTrace();
        }
    }
    /*调用 GoogleAuthSub 来获取 Google 账号的 Authentication Token*/
    private boolean processGoogleLogin(String strUID, String strUPW)
    {
        try
        {
            /*建构自定义的 GoogtleAuthSub 对象*/
            GoogleAuthSub gas = new GoogleAuthSub(strUID, strUPW);
            /*取得 Google Token*/
            String strAuth = gas.getAuthSubToken();
            /*将取回的 Google Token 写入 log 中*/
            Log.i(TAG, strAuth);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return true;
    }
}
```

在上述代码中，方法 showLoginForm()使用 LayoutInflater 获取主 Activity 的 context，并且搭配 AlertDialog 控件构建了一个 Login Form 登录表单。当用户输入账号和密码后，开始重写 DialogInterface. OnClickListener 的 onClick()方法来调用自定义的 processGoogleLogin 处理和 Google 账号验证的连接事件。当通过 Google 验证后取得 Google Authentication Token 后，通过 Intent 打开 example198_01_02.java 以改变 UI 的状态。

（2）编写文件 example_01_02.java，功能是将原来的登录状态改为注销状态，并实现 TextView 的 onClick()方法。文件 example_01_02.java 的具体实现代码如下所示。

```
public class example_01_02 extends Activity
{
    private TextView mTextView03;
    /*中文字的间距*/
    private int intShiftPadding = 14;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loginok);

        /*创建 DisplayMetrics 对象，获取屏幕分辨率*/
        DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(dm);

        /*通过 findViewById()来获取 TextView 对象*/
        mTextView03 = (TextView)findViewById(R.id.myTextView3);
```

```
        /*将文字 Label 放在屏幕右上方*/
        mTextView03.setLayoutParams
        (
          new    AbsoluteLayout.LayoutParams(intShiftPadding*mTextView03.getText().toString().length(),18,(dm.
widthPixels-(intShiftPadding*mTextView03.getText().toString().length()))-10,0)
        );

        /*处理用户单击 TextView 文字的事件处理—注销*/
        mTextView03.setOnClickListener(new TextView.OnClickListener()
        {
          /*覆盖 onClick()方法*/
          @Override
          public void onClick(View v)
          {
            // TODO Auto-generated method stub
            Intent i = new Intent();
            /*注销后调用登录程序(example_01.java)*/
            i.setClass(example_01_02.this, example198.class);
            startActivity(i);
            finish();
          }
        });
      }
}
```

当用户单击 TextView，则通过自定义的 Intent 来调用 example.java，返回到程序的等待状态。

（3）编写文件 GoogleAuthSub.java，此文件是整个实例的核心，通过 Google 提供的 ClientLogin 机制，使用 HttpPost 连接到 https://www.google.com/accounts/ClientLogin，并同时将用户账号和密码及其相关参数以 Name Value Pair 字符串传入，通过自定义方法 getAuth()获取 Google 认证的 Authentication Token，然后模拟 Google 网络服务的 AuthSub 的方法，将自定义的 Header 和 HttpGet 方法带入 Token 来获取用户 Google Calendar 服务中的所有日历数据，并以 XML 文件存储于临时文件中，作为使用 Google 服务的规范。文件 GoogleAuthSub.java 的主要代码如下所示。

```
public class GoogleAuthSub
{
  /*声明变量*/
  private DefaultHttpClient httpclient;
  private HttpPost httpost;
  private HttpResponse response;
  private String strGoogleAccount;
  private String strGooglePassword;
  private String TAG = "IRDC_DEBUG";
  /*GoogleAuthSub 对象构造器*/
  public GoogleAuthSub(String strUID, String strPWD)
  {
    this.strGoogleAccount = strUID;
    this.strGooglePassword = strPWD;
    httpclient = new DefaultHttpClient();
    httpost = new HttpPost("https://www.google.com/accounts/ClientLogin");
  }
  /*取得 Google Token 方法*/
```

```
public String getAuthSubToken()
{
  /*创建 Name Value Pair 字符串*/
  List <NameValuePair> nvps = new ArrayList <NameValuePair>();
  nvps.add(new BasicNameValuePair("Email", this.strGoogleAccount));
  nvps.add(new BasicNameValuePair("Passwd", this.strGooglePassword));
  nvps.add(new BasicNameValuePair("source", "MyApiV1"));
  nvps.add(new BasicNameValuePair("service", "cl"));
  String GoogleLoginAuth="";
  try
  {
    /*创建 Http Post 连接*/
    httpost.setEntity(new UrlEncodedFormEntity(nvps, HTTP.DEFAULT_CONTENT_CHARSET));
    response = httpclient.execute(httpost);
    if( response.getStatusLine().getStatusCode()!=200 )
    {
      return "";
    }
    /*取回 Google Token*/
    InputStream is = response.getEntity().getContent();
    GoogleLoginAuth = getAuth(is);
    /*模拟 HTTP Header*/
    Header[ ] headers = new BasicHeader[6];
    headers[0] = new BasicHeader("Content-type", "application/x-www-form-urlencoded");
    headers[1] = new BasicHeader("Authorization", "GoogleLogin auth=\""+GoogleLoginAuth+"\"");
    headers[2] = new BasicHeader("User-Agent", "Java/1.5.0_06");
    headers[3] = new BasicHeader("Accept", "text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2");
    headers[4] = new BasicHeader("Connection", "keep-alive");
    /*发出 Http Get 请求登录 Google Calendar 服务作范例*/
    HttpGet httpget;
    String feedUrl2 = "http://www.google.com/calendar/feeds/default/allcalendars/full";
    httpget = new HttpGet(feedUrl2);
    httpget.addHeader(headers[0]);
    httpget.addHeader(headers[1]);
    httpget.addHeader(headers[2]);
    httpget.addHeader(headers[3]);
    httpget.addHeader(headers[4]);
    /*取得 Google Calendar 服务应答*/
    response = httpclient.execute(httpget);
    String strTemp01 = convertStreamToString(response.getEntity().getContent());
    Log.i(TAG, strTemp01);
    /*指定暂存盘位置*/
    String strEarthLog = "/sdcard/googleauth.log";
    BufferedWriter bw;
    bw = new BufferedWriter (new FileWriter(strEarthLog));
    /*将取回文件写入暂存盘中*/
    bw.write( strTemp01, 0, strTemp01.length() );
    bw.flush();

  }
  catch (UnsupportedEncodingException e)
```

```
    {
        e.printStackTrace();
    }
    catch (ClientProtocolException e)
    {
        e.printStackTrace();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return GoogleLoginAuth;
}
/*自定义读取 Token 内容的方法*/
public String getAuth(InputStream is)
{
    BufferedReader reader = new BufferedReader(new InputStreamReader(is));
    String line = null;
    String strAuth="";
    try
    {
        while ((line = reader.readLine()) != null)
        {
            Log.d(TAG, ": "+line);
            if( line.startsWith("Auth="))
            {
                strAuth=line.substring(5);
                Log.i("auth",": "+strAuth);
            }
        }
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
    finally
    {
        try
        {
            is.close();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
    return strAuth;
```

```
    }
    /*将数据转为字符串方法*/
    public String convertStreamToString(InputStream is)
    {
        BufferedReader reader = new BufferedReader(new InputStreamReader(is));
        StringBuilder sb = new StringBuilder();
        String line = null;
        try
        {
            while ((line = reader.readLine()) != null)
            {
                sb.append(line);
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        finally
        {
            try
            {
                is.close();
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
        }
        return sb.toString();
    }
}
```

执行后的效果如图 13-22 所示；单击“登录”超链接后显示登录表单界面，如图 13-23 所示；输入账号和密码并单击 OK 按钮后弹出“成功获取 Token!!”提示，如图 13-24 所示。



图 13-22　执行效果



图 13-23　登录表单

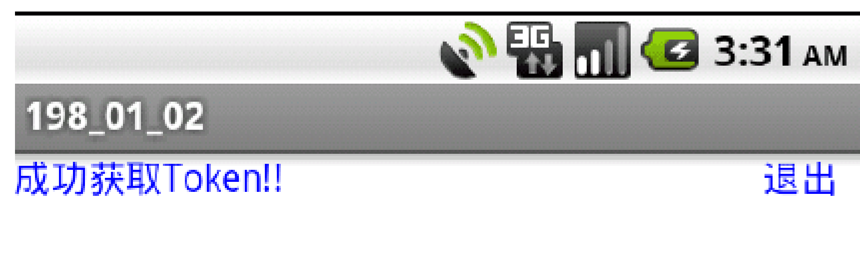


图 13-24　“成功获取 Token”提示

13.8　实现谷歌搜索功能

实例 153	在手机中实现谷歌搜索功能
源码路径	光盘:\daima\153
视频路径	光盘:\视频\153
实例必备	153.使用 Google Search API 的流程.pdf

13.8.1　实例说明

在现实信息时代，信息已经成为第一生产力。在巨大的网络资源中，检索站点蓬勃发展，百度、雅虎和 Google 都已经站在了时代的最前沿。在 Android 官方服务中，提供了 Google Search API 实现检索处理。在本实例中，将演示使用 Google Search API 实现检索处理的方法。

13.8.2　具体实现

（1）编写文件 example.java，在此创建了一个 MyAdapter 对象，此对象是自己实现的 BaseMyAdapter 类。文件 example.java 的主要实现代码如下所示。

```
public class example extends Activity
{
  private AutoCompleteTextView myAutoCompleteTextView1;
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    myAutoCompleteTextView1 = (AutoCompleteTextView) findViewById(R.id.myAutoCompleteTextView1);
    /*new 一个自己实现的 BaseAdapter*/
    MyAdapter adapter = new MyAdapter(this);
    myAutoCompleteTextView1.setAdapter(adapter);
  }
}
```

（2）编写文件 MyAdapter.java，定义了继承于 BaseAdapter 的类 MyAdapter，可以通过覆盖 Filterable 对象中的 getFilter()方法来动态处理输入的关键字。当用户输入关键字时，方法 performFiltering()所返回的 FilterResults 就是查询后的结果。文件 MyAdapter.java 的主要实现代码如下所示。

```
public class MyAdapter extends BaseAdapter implements Filterable
{
  ArrayList<String> keyWordValue = new ArrayList<String>();
  ArrayList<String> resultValue = new ArrayList<String>();
  private Context mContext;
  LinearLayout.LayoutParams param1;
  public MyAdapter(Context context)
  {
```

```
        mContext = context;
        param1 = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
    }
    public int getCount()
    {
        return keyWordValue.size();
    }
    public Object getItem(int position)
    {
        return keyWordValue.get(position);
    }
    public long getItemId(int position)
    {
        return position;
    }
    public View getView(int position, View view, ViewGroup viewGroup)
    {
        LinearLayout myLinearLayout = new LinearLayout(mContext);
        myLinearLayout.setOrientation(LinearLayout.HORIZONTAL);
        if (position >= keyWordValue.size())
            return myLinearLayout;
        /*第一个 TextView 放关键字*/
        TextView keyWordTextView = new TextView(this.mContext);
        keyWordTextView.setTextColor(mContext.getResources().getColor(
                R.drawable.blue));
        keyWordTextView.setTextSize(18);
        keyWordTextView.setWidth(180);
        try
        {
            keyWordTextView
                    .setText(keyWordValue.get(position).toString());
        } catch (java.lang.IndexOutOfBoundsException i)
        {
            keyWordTextView.setText("");
        }
        /*第二个 TextView 放关键字结果数量*/
        TextView resultTextView = new TextView(this.mContext);
        resultTextView.setTextColor(mContext.getResources().getColor(
                R.drawable.red));
        resultTextView.setTextSize(18);
        try
        {
            resultTextView.setText(resultValue.get(position).toString());
        } catch (java.lang.IndexOutOfBoundsException i)
        {
            resultTextView.setText("");
        }
        myLinearLayout.addView(keyWordTextView, param1);
        myLinearLayout.addView(resultTextView, param1);
```

```
    return myLinearLayout;
}
public Filter getFilter()
{
    // TODO Auto-generated method stub
    Filter myFilter = new Filter()
    {
        protected FilterResults performFiltering(CharSequence text)
        {
            FilterResults fr = new FilterResults();
            keyWordValue = new java.util.ArrayList<String>();
            resultValue = new java.util.ArrayList<String>();
            if (text == null || text.length() == 0)
            {
                fr.count = keyWordValue.size();
                fr.values = keyWordValue;
                return fr;
            }
            /*输入关键字后调用 Google 关键字 API*/
            changeResult(getGoogleAPI(text.toString()));
            fr.count = keyWordValue.size();
            fr.values = keyWordValue;
            return fr;
        }
        protected void publishResults(CharSequence text,
                FilterResults filterResults)
        {
            if (filterResults != null && filterResults.count > 0)
                notifyDataSetChanged();
            else
                notifyDataSetInvalidated();
        }
    };
    return myFilter;
}
/*访问 GoogleAPI 取得返回的结果字符串*/
private String getGoogleAPI(String text)
{
    String uri = "";
    try
    {
        /*输入的字要编码*/
        uri = "http://www.google.com/complete/"
                + "search?hl=en&js=true&qu="
                + URLEncoder.encode(text, "utf-8");
    } catch (UnsupportedEncodingException e1)
    {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    URL googleUrl = null;
```

```
        HttpURLConnection conn = null;
        InputStream is = null;
        BufferedReader in = null;
        String resultStr = "";
        /*取得连接*/
        try
        {
          googleUrl = new URL(uri);
          /*打开连接*/
          conn = (HttpURLConnection) googleUrl.openConnection();
          int code = conn.getResponseCode();
          /*连接 OK 时*/
          if (code == HttpURLConnection.HTTP_OK)
          {
            /*取得返回的 InputStream*/
            is = conn.getInputStream();
            in = new BufferedReader(new InputStreamReader(is));
            String inputLine;
            /*一行一行读取*/
            while ((inputLine = in.readLine()) != null)
            {
              resultStr += inputLine;
            }
          }
        } catch (IOException e)
        {
          // TODO Auto-generated catch block
          e.printStackTrace();
        } finally
        {
          try
          {
            if (is != null)
              is.close();
            if (conn != null)
              conn.disconnect();
          } catch (Exception e)
          {
          }
        }
        return resultStr;
      }
      /*处理返回的字符串变成 ArrayList*/
      private void changeResult(String text)
      {
        String resultStr = "";
        String startSub = "new Array(2, ";
        String endSub = "), new Array";
        int start = text.indexOf(startSub);
        int end = text.indexOf(endSub);
        if (start != -1 && end != -1)
```

```
        {
            resultStr = text.substring(start + startSub.length(), end);
            /*去掉前后的 “"” */
            resultStr = resultStr.substring(1, resultStr.length() - 1);
            /*以 “,” 来分隔字符串变成字符串数组*/
            String total[ ] = resultStr.split("\\\", \\\"");
            for (int i = 0; i < total.length / 2; i++)
            {
                keyWordValue.add(total[i * 2]);
                /*将 results 字符串去掉*/
                resultValue
                        .add(total[i * 2 + 1].replaceAll(" results", ""));
            }
        }
    }
}
```

在上述 MyAdapter 类中重写了 getView()，在其中放了两个 TextView 控件，一个用于显示关键字，另一个用于显示结果数量。因为 AutoCompleteTextView 绑定了 MyAdapter，所以当 Adapter 动态向 Google 获取查询结果时，也顺便更新了 AutoCompleteTextView 下拉菜单中的结果。

执行后的效果如图 13-25 所示。

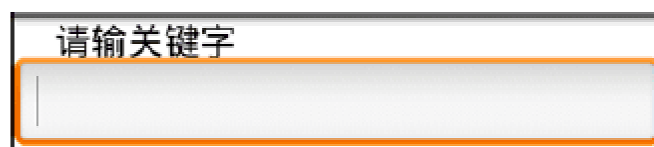


图 13-25　执行效果

13.9　使用 Google Chart API 生成二维条码

实例 154	使用 Google Chart API 生成二维条码
源码路径	光盘:\daima\154
视频路径	光盘:\视频\154
实例必备	154.Google Chart API 的用法详解.pdf

13.9.1　实例说明

在 Android 系统中，使用 Google Chart API 可以方便地生成动态二维条码。这样开发人员无需掌握 GD Libray 等知识，降低了开发门槛。通过本实例的实现过程，详细介绍了在 Android 中通过 Google Chart API 动态生成二维条码的具体流程。

13.9.2　具体实现

在文件 example.java 中，通过自定义函数 genGoogleQRChart 生成要显示远程图像的网址，然后使用<img src="">的方式组成 HTML 标签。在实现成像功能时，通过 WebView 来显示 HTML 的内容。文

件 example.java 的主要实现代码如下所示。

```
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
/*测试手机是否具有连接 Google API 的连接能力*/
if(checkInternetConnection
    ("http://code.google.com/intl/zh-TW/apis/chart/","utf-8")
  )
{
  bInternetConnectivity = true;
}
mWebView01 = (WebView)findViewById(R.id.myWebView1);
mButton01 = (Button)findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View v)
  {
    // TODO Auto-generated method stub
    if(mEditText01.getText().toString()!="" &&
        bInternetConnectivity==true)
    {
      mWebView01.loadData
      (
        /*调用自定义云端生成 QR Code 函数*/
        genGoogleQRChart
        (
          mEditText01.getText().toString(),120
        ),"text/html", "utf-8"
      );
    }
  }
});
mEditText01 = (EditText)findViewById(R.id.myEditText1);
mEditText01.setText(R.string.str_text);

mEditText01.setOnKeyListener(new EditText.OnKeyListener()
{
  @Override
  public boolean onKey(View v, int keyCode, KeyEvent event)
  {
    if(mEditText01.getText().toString()!="" &&
        bInternetConnectivity==true)
    {
      mWebView01.loadData
      (
        genGoogleQRChart
        (
          mEditText01.getText().toString(),120
        ),"text/html", "utf-8"
      );
    }
```

```
            return false;
        }
    });
/*调用 Google API，产生 QR Code 二维条形码*/
public String genGoogleQRChart(String strToQRCode, int strWidth)
{
    String strReturn="";
    try
    {
        strReturn = new String(strToQRCode.getBytes("utf-8"));
        /*组成 Google API 需要的传输参数字符串*/
        strReturn = "<html><body>"+
        "<img src=http://chart.apis.google.com/chart?chs="+
        strWidth+"x"+strWidth+"&chl="+
        URLEncoder.encode(strReturn, "utf-8")+
        "&choe=UTF-8&cht=qr></body></html>";
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return strReturn;
}
/*检查网络连接是否正常*/
public boolean checkInternetConnection
(String strURL, String strEncoding)
{
    /*最多延时 n 秒，若无应答则表示无法连接*/
    int intTimeout = 5;
    try
    {
        HttpURLConnection urlConnection= null;
        URL url = new URL(strURL);
        urlConnection=(HttpURLConnection)url.openConnection();
        urlConnection.setRequestMethod("GET");
        urlConnection.setDoOutput(true);
        urlConnection.setDoInput(true);
        urlConnection.setRequestProperty
        (
            "User-Agent","Mozilla/4.0"+
            " (compatible; MSIE 6.0; Windows 2000)"
        );
        urlConnection.setRequestProperty
        ("Content-type","text/html; charset="+strEncoding);
        urlConnection.setConnectTimeout(1000*intTimeout);
        urlConnection.connect();
        if (urlConnection.getResponseCode() == 200)
        {
            return true;
        }
        else
```

```
            {
                return false;
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return false;
        }
    }
    /*自定义 biG5 转 UTF-8*/
    public String big52unicode(String strBIG5)
    {
        String strReturn="";
        try
        {
            strReturn = new String(strBIG5.getBytes("big5"), "UTF-8");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return strReturn;
    }

    /*自定义 UTF-8 转 biG5*/
    public String unicode2big5(String strUTF8)
    {
        String strReturn="";
        try
        {
            strReturn = new String(strUTF8.getBytes("UTF-8"), "big5");
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        return strReturn;
    }
```

执行后的效果如图 13-26 所示。



图 13-26　执行效果

13.10　在手机中编写一个翻译软件

实例 155	在手机中编写一个翻译软件
源码路径	光盘:\daima\155
视频路径	光盘:\视频\155
实例必备	155.Ajax 语言的 API.pdf

13.10.1　实例说明

Android 系统中不具备手机翻译功能，但是使用 Google Translate API 可以编写具有翻译功能的程序。本实例将详细介绍在 Android 中通过 Google Translate API 实现翻译处理的过程。

13.10.2　具体实现

本实例的主文件是 example.java，其具体实现流程如下所示。

（1）在 Activity 中设置一个 EditText Widget，用于接收用户欲翻译的字符串。

（2）编写和 Google Translate API 通信的 JavaScript，并以 HTML 格式存储在 assets 文件夹中。

（3）在 HTML 网页中编写一个<a href>超链接。

（4）当用户在 EditText 输入英文后，单击“中文”超链接后开始翻译处理，并将翻译结果显示在 WebView 中。

文件 example.java 的主要实现代码如下所示。

```
    myEditText1 = (EditText) findViewById(R.id.myEditText1);
    myWebView1 = (WebView) findViewById(R.id.myWebView1);
    /*获取 WebSettings*/
    WebSettings webSettings = myWebView1.getSettings();
    /*设置可运行 JavaScript*/
    webSettings.setJavaScriptEnabled(true);
    webSettings.setSaveFormData(false);
    webSettings.setSavePassword(false);
    webSettings.setSupportZoom(false);
    myWebView1.setWebChromeClient(new MyWebChromeClient());
    /*设置给 HTML 调用的对象及名称*/
    myWebView1.addJavascriptInterface(new runJavaScript(), "irdc");
    /*将 assets/google_translate.html 载入*/
    String url = "file:///android_asset/google_translate.html";
    myWebView1.loadUrl(url);
  ……
  final class runJavaScript
  {
    public void runOnAndroidJavaScript()
    {
      mHandler01.post(new Runnable()
```

```
        {
            public void run()
            {
                if (myEditText1.getText().toString() != "")
                {
                    /*调用 google_translate.html 中的 javascript*/
                    myWebView1.loadUrl("javascript:translate('"
                        + myEditText1.getText().toString() + "')");
                }
            }
        });
    }
}
/*捕捉网页中的 alert javascript 作为.js 调试之用，并输出至 LogCat*/
final class MyWebChromeClient extends WebChromeClient
{
    @Override
    public boolean onJsAlert(WebView view, String url,
            String message, JsResult result)
    {
        Log.d(LOG_TAG, message);
        return super.onJsAlert(view, url, message, result);
    }
}
```

执行后的效果如图 13-27 所示。输入英文字符并单击“中文”超链接后将会显示翻译结果。例如，输入 name 后翻译结果如图 13-28 所示。



图 13-27　执行效果

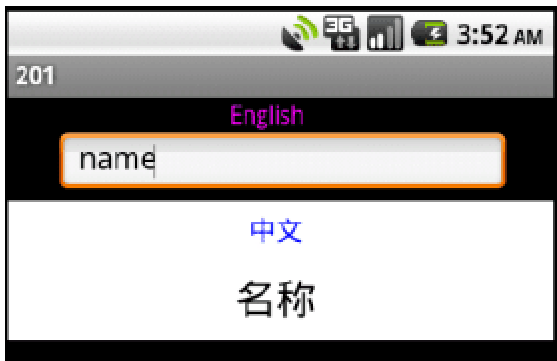


图 13-28　翻译结果

13.11　在手机屏幕中生成二维条码

实例 156	在手机屏幕中生成二维条码
源码路径	光盘:\daima\156
视频路径	光盘:\视频\156
实例必备	156.DisplayMetircs 处理分辨率问题.pdf

13.11.1　实例说明

在本章前面的实例中介绍过使用 Google Chat API 生成二维条码的流程，但是不能保证手机都处于

联网状态，也就不能确保一定可以使用 Google Chat API。为了解决上述问题，可以使用开放的资源来实现这个功能，例如，最常见的开发资源是 swetake，读者可以从 http://www.swetake.com/下载。下载后将其引入 Android 工程中，并将文件名称改为 SwetakeQRCode.jar。

13.11.2　具体实现

编写文件 example.java，其具体实现流程如下所示。

（1）设置应用程序全屏幕运行，而不使用 title 标签，具体代码如下所示。

```
/*使应用程序全屏幕运行，不使用 title 标签*/
requestWindowFeature(Window.FEATURE_NO_TITLE);
setContentView(R.layout.main);
```

（2）取得屏幕解析像素，并以 SurfaceView 作为相机 Preview 之用，绑定 SurfaceView，取得 SurfaceHolder 对象，并产生 QRCode 的按钮事件处理，具体代码如下所示。

```
/*取得屏幕解析像素*/
DisplayMetrics dm = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(dm);

mTextView01 = (TextView) findViewById(R.id.myTextView1);
mTextView01.setText(R.string.str_qr_gen);

/*以 SurfaceView 作为相机 Preview 之用*/
mSurfaceView01 = (SurfaceView) findViewById(R.id.mSurfaceView1);

/*绑定 SurfaceView，取得 SurfaceHolder 对象*/
mSurfaceHolder01 = mSurfaceView01.getHolder();

/*Activity 必须实现 SurfaceHolder.Callback*/
mSurfaceHolder01.addCallback(example198.this);

/*产生 QRCode 的按钮事件处理*/
mButton01 = (Button)findViewById(R.id.myButton1);
mButton01.setOnClickListener(new Button.OnClickListener()
{
  @Override
  public void onClick(View arg0)
  {
    // TODO Auto-generated method stub
    if(mEditText01.getText().toString()!="")
    {
      /*传入 setQrcodeVersion 为 4，仅能接受 62 个字符*/
      AndroidQREncode(mEditText01.getText().toString(), 4);
    }
  }
});

mEditText01 = (EditText)findViewById(R.id.myEditText1);
mEditText01.setText("DavidLanz");
mEditText01.setOnKeyListener(new EditText.OnKeyListener()
```

```
    {
        @Override
        public boolean onKey(View v, int keyCode, KeyEvent event)
        {
            return false;
        }
    });
```

（3）自定义产生 QR Code 函数 AndroidQREncode 以实现二维编码处理，具体代码如下所示。

```
/*自定义产生 QR Code 的函数*/
public void AndroidQREncode(String strEncoding, int qrcodeVersion)
{
    try
    {
        /*构建 QRCode 编码对象*/
        com.swetake.util.Qrcode testQrcode =
        new com.swetake.util.Qrcode();
        testQrcode.setQrcodeErrorCorrect('M');
        testQrcode.setQrcodeEncodeMode('B');
        testQrcode.setQrcodeVersion(qrcodeVersion);
        byte[ ] bytesEncoding = strEncoding.getBytes("utf-8");
        if (bytesEncoding.length>0 && bytesEncoding.length <120)
        {
            /*将字符串通过 calQrcode 函数转换成 boolean 数组*/
            boolean[ ][ ] bEncoding = testQrcode.calQrcode(bytesEncoding);
            /*依据编码后的 boolean 数组绘图*/
            drawQRCode
            (bEncoding, getResources().getColor(R.drawable.black));
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

（4）定义方法 drawQRCode()，先在 SurfaceView 上绘制 QR Code 条形码，然后解锁 SurfaceHolder 并绘图，具体代码如下所示。

```
/*在 SurfaceView 上绘制 QR Code 条形码*/
private void drawQRCode(boolean[ ][ ] bRect, int colorFill)
{
    /*test Canvas*/
    int intPadding = 20;

    /*欲在 SurfaceView 上绘图，需先锁定 SurfaceHolder*/
    Canvas mCanvas01 = mSurfaceHolder01.lockCanvas();

    /*设置画布绘制颜色*/
    mCanvas01.drawColor(getResources().getColor(R.drawable.white));

    /*创建画笔*/
    Paint mPaint01 = new Paint();
```

```
    /*设置画笔颜色及模式*/
    mPaint01.setStyle(Paint.Style.FILL);
    mPaint01.setColor(colorFill);
    mPaint01.setStrokeWidth(1.0F);

    /*逐一加载二维 boolean 数组*/
    for (int i=0;i<bRect.length;i++)
    {
      for (int j=0;j<bRect.length;j++)
      {
        if (bRect[j][i])
        {
          /*依据数组值，绘出条形码方块*/
          mCanvas01.drawRect
          (
            new Rect
            (
              intPadding+j*3+2,
              intPadding+i*3+2,
              intPadding+j*3+2+3,
              intPadding+i*3+2+3
            ), mPaint01
          );
        }
      }
    }
    /*解锁 SurfaceHolder 并绘图*/
    mSurfaceHolder01.unlockCanvasAndPost(mCanvas01);
  }

  public void mMakeTextToast(String str, boolean isLong)
  {
    if(isLong==true)
    {
      Toast.makeText(example198.this, str, Toast.LENGTH_LONG).show();
    }
    else
    {
      Toast.makeText(example198.this, str, Toast.LENGTH_SHORT).show();
    }
  }
  @Override
  public void surfaceChanged
  (SurfaceHolder surfaceholder, int format, int w, int h)
  {
    Log.i(TAG, "Surface Changed");
  }
  @Override
  public void surfaceCreated(SurfaceHolder surfaceholder)
  {
```

```
        Log.i(TAG, "Surface Changed");
    }
    public void surfaceDestroyed(SurfaceHolder surfaceholder)
    {
        Log.i(TAG, "Surface Destroyed");
    }
}
```

执行后可以将输入的文本字符转换为二维条形码，如图 13-29 所示。



图 13-29　执行效果

第 14 章　传感器实战应用

Android 系统中提供的主要传感器有加速度传感器、磁场传感器、方向传感器、陀螺仪传感器、光线传感器、压力传感器、温度传感器和距离传感器等。传感器系统会主动对上层报告传感器精度和数据的变化，并且提供了设置传感器精度的接口，这些接口可以在 Java 应用和 Java 框架中使用。本章将通过各个传感器实例的实现过程，详细讲解在 Android 系统中使用传感器技术的基本流程。

14.1　检测当前设备支持的传感器

实例 157	检测当前设备支持的传感器
源码路径	光盘:\daima\157
视频路径	光盘:\视频\157
实例必备	157.Android 传感器系统概述.pdf ① 传感器系统的 Java 部分 ② 传感器系统的 JNI 部分 ③ 传感器系统 HAL 层 ④ 驱动层

14.1.1　实例说明

版本 Android 4.4 中共提供了 18 种传感器 API，各个类型的具体说明如下所示。

（1）TYPE_ACCELEROMETER：加速度传感器，单位是 m/s^2，测量应用于设备 x、y、z 轴上的加速度，又叫做 G-sensor。

（2）TYPE_AMBIENT_TEMPERATURE：温度传感器，单位是℃，能够测量并返回当前的温度。

（3）TYPE_GRAVITY：重力传感器，单位是 m/s^2，用于测量设备 x、y、z 轴上的重力，又叫做 GV-sensor，地球上的数值是 $9.8m/s^2$，也可以设置其他星球上的值。

（4）TYPE_GYROSCOPE：陀螺仪传感器，单位是 rad/s，能够测量设备 x、y、z 轴的角加速度数据。

（5）TYPE_LIGHT：光线传感器，单位是 lx，能够检测周围的光线强度，在手机系统中主要用于调节 LCD 亮度。

（6）TYPE_LINEAR_ACCELERATION：线性加速度传感器，单位是 m/s^2，能够获取加速度传感器去除重力的影响得到的数据。

（7）TYPE_MAGNETIC_FIELD：磁场传感器，单位是 μT（微特斯拉），能够测量设备周围 3 个

物理轴（x，y，z）的磁场。

（8）TYPE_ORIENTATION：方向传感器，用于测量设备围绕 3 个物理轴（x，y，z）的旋转角度，在新版本中已经使用 SensorManager.getOrientation()替代。

（9）TYPE_PRESSURE：压力传感器，单位是 hPa（百帕斯卡），能够返回当前环境下的压强。

（10）TYPE_PROXIMITY：距离传感器，单位是 cm，能够测量某个对象到屏幕的距离。可以在打电话时判断人耳到电话屏幕的距离，以关闭屏幕而达到省电的目的。

（11）TYPE_RELATIVE_HUMIDITY：湿度传感器，能够测量周围环境的相对湿度。

（12）TYPE_ROTATION_VECTOR：旋转向量传感器，旋转矢量代表设备的方向，是一个将坐标轴和角度混合计算得到的数据。

（13）TYPE_TEMPERATURE：温度传感器，在新版本中被 TYPE_AMBIENT_TEMPERATURE 替换。

（14）TYPE_ALL：返回所有的传感器类型。

（15）TYPE_GAME_ROTATION_VECTOR：除了不能使用地磁场之外，和 TYPE_ROTATION_VECTOR 的功能完全相同。

（16）TYPE_GYROSCOPE_UNCALIBRATED：提供了能够让应用调整传感器的原始值，定义了一个描述未校准陀螺仪的传感器类型。

（17）TYPE_MAGNETIC_FIELD_UNCALIBRATED：和 TYPE_GYROSCOPE_UNCALIBRATED 相似，也提供了能够让应用调整传感器的原始值，定义了一个描述未校准陀螺仪的传感器类型。

（18）TYPE_SIGNIFICANT_MOTION：运动触发传感器，应用程序不需要为这种传感器触发任何唤醒锁。能够检测当前设备是否运动，并发送检测结果。

14.1.2 具体实现

（1）布局文件 main.xml 的具体实现代码如下所示。

```
<linearlayout android:layout_height="fill_parent" android:layout_width="fill_parent" android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android">
<textview android:layout_height="wrap_content"
 android:layout_width="fill_parent" android:text=""
 android:id="@+id/TextView01"
>
</textview>
</linearlayout>
```

（2）主程序文件 MainActivity.java 的具体实现代码如下所示。

```
public class MainActivity extends Activity {

	/** Called when the activity is first created.*/
	@SuppressWarnings("deprecation")
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		//准备显示信息的 UI 组件
```

```
        final TextView tx1 = (TextView) findViewById(R.id.TextView01);

        //从系统服务中获得传感器管理器
        SensorManager sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

        //从传感器管理器中获得全部的传感器列表
        List<Sensor> allSensors = sm.getSensorList(Sensor.TYPE_ALL);

        //显示有多少个传感器
        tx1.setText("经检测该手机有" + allSensors.size() + "个传感器，分别是：\n");

        //显示每个传感器的具体信息
        for (Sensor s : allSensors) {

            String tempString = "\n" + "  设备名称：" + s.getName() + "\n" + "  设备版本：" +
s.getVersion() + "\n" + "  供应商："
                    + s.getVendor() + "\n";

            switch (s.getType()) {
            case Sensor.TYPE_ACCELEROMETER:
                tx1.setText(tx1.getText().toString() + s.getType() + " 加速度传感器 accelerometer"
+ tempString);
                break;
            case Sensor.TYPE_GYROSCOPE:
                tx1.setText(tx1.getText().toString() + s.getType() + " 陀螺仪传感器 gyroscope"
+ tempString);
                break;
            case Sensor.TYPE_LIGHT:
                tx1.setText(tx1.getText().toString() + s.getType() + " 光线传感器 light" +
tempString);
                break;
            case Sensor.TYPE_MAGNETIC_FIELD:
                tx1.setText(tx1.getText().toString() + s.getType() + " 磁场传感器 magnetic field"
+ tempString);
                break;
            case Sensor.TYPE_ORIENTATION:
                tx1.setText(tx1.getText().toString() + s.getType() + " 方向传感器 orientation" +
tempString);
                break;
            case Sensor.TYPE_PRESSURE:
                tx1.setText(tx1.getText().toString() + s.getType() + " 压力传感器 pressure"
+ tempString);
                break;
            case Sensor.TYPE_PROXIMITY:
                tx1.setText(tx1.getText().toString() + s.getType() + " 距离传感器 proximity"
+ tempString);
                break;
            case Sensor.TYPE_AMBIENT_TEMPERATURE :
                tx1.setText(tx1.getText().toString() + s.getType() + " 温度传感器 temperature" +
tempString);
                break;
```

```
                        default:
                                tx1.setText(tx1.getText().toString() + s.getType() + "  未知传感器" + tempString);
                                break;
                        }
                }

        }
}
```

上述实例代码需要在真机中运行，执行后将会列表显示当前设备所支持的传感器类型，如图 14-1 所示。



图 14-1　传感器测试

14.2　获取设备中光线传感器的值

实例 158	获取设备中光线传感器的值
源码路径	光盘:\daima\158
视频路径	光盘:\视频\158
实例必备	158.光线传感器基础.pdf ① 光线传感器介绍 ② 在 Android 中使用光线传感器的方法

14.2.1　实例说明

光线传感器的好处是可以根据手机所处环境的光线来调节手机屏幕的亮度和键盘灯。例如，在光线充足的地方屏幕会很亮，键盘灯就会关闭。相反，如果在暗处，键盘灯就会亮，屏幕较暗（与屏幕亮度的设置也有关系），这样既保护了眼睛，又节省了电量。光线传感器在进入睡眠模式时会发出蓝色

周期性闪动的光，非常美观。本实例中演示了在 Android 设备中使用光线传感器的方法。

14.2.2　具体实现

（1）编写布局文件 activity_main.xml，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

（2）编写程序文件 MainActivity.java，具体实现代码如下所示。

```
package com.example.sensor;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.renderscript.Sampler.Value;
import android.app.Activity;
import android.view.Menu;
import android.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener   {



    private SensorManager sensor;
    private TextView text;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensor = (SensorManager)getSystemService(SENSOR_SERVICE);
        text = (TextView)findViewById(R.id.textView1);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

```
            // Inflate the menu; this adds items to the action bar if it is present
            getMenuInflater().inflate(R.menu.activity_main, menu);
            return true;
      }

      @Override
      protected void onPause() {
            // TODO Auto-generated method stub
            sensor.unregisterListener(this);
            super.onPause();
      }

      @Override
      protected void onResume() {
            // TODO Auto-generated method stub

      sensor.registerListener(this,sensor.getDefaultSensor(Sensor.TYPE_LIGHT),SensorManager.SENSOR_
DELAY_GAME);
            super.onResume();
      }

      @Override
      protected void onStop() {
            // TODO Auto-generated method stub
            sensor.unregisterListener(this);
            super.onStop();
      }



      @Override
      public void onAccuracyChanged(Sensor sensor, int accuracy) {
            // TODO Auto-generated method stub

      }

      @Override
      public void onSensorChanged(SensorEvent event) {
            // TODO Auto-generated method stub
            float[ ] values = event.values;
            int sensorType = event.sensor.TYPE_LIGHT;
            if(sensorType==Sensor.TYPE_LIGHT)
            {
                  text.setText(String.valueOf(values[0]));
            }
      }
}
```

在真机中执行后，将会显示设备中光线传感器的值。

14.3　在设备地图中快速查询某个位置

实例 159	在设备地图中快速查询某个位置
源码路径	光盘:\daima\159
视频路径	光盘:\视频\159
实例必备	159.类 Geocoder 的继承关系.pdf

14.3.1　实例说明

在现实世界中，地图和定位服务通常使用经纬度来精确地指出地理位置。在 Android 系统中，提供了地理编码类 Geocoder 来转换经纬度和现实世界的地址。地理编码是一个街道、地址或者其他位置（经度、纬度）转换为坐标的过程。反向地理编码是将坐标转换为地址（经度、纬度）的过程。一组反向地理编码结果间可能会有所差异。例如，在一个结果中可能包含最临近建筑的完整街道地址，而另一个可能只包含城市名称和邮政编码。Geocoder 要求的后端服务并没有包含在基本的 Android 框架中。如果没有此后端服务，执行 Geocoder 的查询方法将返回一个空列表。使用 isPresent()方法，以确定 Geocoder 是否能够正常执行。

14.3.2　具体实现

（1）编写文件 mymap.xml，在其中添加 Map 密钥以实现对地图的引用，主要代码如下所示。

```
<com.google.android.maps.MapView
                        android:id="@+id/mapview_mymap_display"
                        android:layout_width="fill_parent"
                        android:layout_height="fill_parent"
                        android:apiKey="0NFa8R5kt6KmenQdcxhltm2rcaSZaNhOe3WZQTw"
                        />
```

（2）编写文件 MyMap.java，获取用户在文本框中输入的地址，并根据这个地址进行查询操作。文件 MyMap.java 的主要代码如下所示。

```
public class MyMap extends MapActivity{//程序列表中要添加联网的权限还要加一个类库

     MapView mapview;
    private MapController mapcontroller;
    private GeoPoint geopoint;
    protected String addressname;

     protected void onCreate(Bundle savedInstanceState) {
           // TODO Auto-generated method stub
           super.onCreate(savedInstanceState);
           setContentView(R.layout.mymap);
           //用于显示地图上的一个 ViewGroup
           mapview=(MapView)findViewById(R.id.mapview_mymap_display);
```

```
        Bundle bundle=getIntent().getExtras();
        Log.d("MyMap_Oncreate_bundle",bundle+"");
        addressname=bundle.getString("address");
        Log.d("MyMap_oncreate",addressname);
        //使得这个 view 可以获得单击事件
        mapview.setClickable(true);
        //是否可以设置自动缩放
        mapview.setBuiltInZoomControls(true);

        //获取控制缩放的操作对象
        mapcontroller=mapview.getController();
        //通过系统默认区域设置进行地图的定位
        Geocoder geocoder=new Geocoder(this);

        mapview.setTraffic(true);
        try
        {
                List<Address> addresses=geocoder.getFromLocationName(addressname,2);
                Log.d("MyMap_oncreate_addressname3",addressname);
                geopoint = new GeoPoint(
                        (int) (addresses.get(0).getLatitude() * 1E6),
                        (int) (addresses.get(0).getLongitude() * 1E6));
                MyOverlay myoverlay=new MyOverlay();

                mapview.getOverlays().add(myoverlay);
                mapcontroller.setZoom(20);
                mapcontroller.animateTo(geopoint);
        }
        catch(Exception e)
        {
                e.printStackTrace();
        }
}
@Override
protected boolean isRouteDisplayed() {          return false;
}

class MyOverlay extends Overlay
{
        @Override
        public boolean draw(Canvas canvas, MapView mapview, boolean shadow, long when) {
                // TODO Auto-generated method stub
                Paint paint=new Paint();
                Point screenPoint=new Point();
                //经纬度坐标和屏幕像素坐标的一个映射
                mapview.getProjection().toPixels(geopoint, screenPoint);
                //并且这个映射可以把地理上的经纬度转换成屏幕上的像素点
                Bitmap bitmap=BitmapFactory.decodeResource(getResources(),R.drawable.flag1);
                canvas.drawBitmap(bitmap,screenPoint.x,screenPoint.y, paint);
```

```
                    canvas.drawText(addressname,screenPoint.x,screenPoint.y, paint);
                    return super.draw(canvas, mapview, shadow, when);
              }
        }
}
```

执行后的效果如图 14-2 所示，输入一个地址并单击“开始查询”按钮后会在地图中显示其位置。

在本实例中，在 Activity 的 OnCreate()方法中设置 MapView 的各个属性，例如，设置了是否可以获得单击事件（setClickable()方法），设置了地图缩放尺度（setBuiltInZoomControls(true)），设置了地图的视图模式，谷歌地图一共有 3 种视图模式。



图 14-2　执行效果

- ☑　街道视图：mapview.setStreetView()。
- ☑　卫星视图：mapview.setSatelite()。
- ☑　一般地图：mapview.setTraffic()。

对地图的操作是通过对一个 MapController 对象的操作实现的，该对象通过 MapView.getController()方法获取。在使地图显示某一个地点时，则使用 MapController.animateTo()方法，参数是一个 GeoPoint 类型，经度和纬度的一个组合，类似于坐标值，并且可以通过 MapController.setZoom 来设置放大的倍数，其中数值越大，地图越详细。Geocoder 类是处理地理编码的一个类，根据输入的地点可以获取一个和此地点相关的 Address 类的集合。

getFromLocationName()方法有两个参数，一个是输入的地点，另一个是获取的地点的个数。也可通过继承 OverLay 类为地图设置一个图标图层。

最后不要忘记在列表中添加访问 Internet 的权限。

```
<uses-permission android:name="android:permission.Internet"/>
```

还需要为应用添加类库。

```
<uses- library android:name="com.google.android.maps"/>
```

14.4　获取磁场传感器的 3 个分量

实例 160	获取磁场传感器的 3 个分量
源码路径	光盘:\daima\160
视频路径	光盘:\视频\160
实例必备	160.磁场传感器基础.pdf ① 什么是磁场传感器 ② 磁场传感器的分类

14.4.1　实例说明

在市面中，最早磁场传感器是伴随测磁仪器的进步而逐步发展的。在众多的测试磁场方法中，大多都是将磁场信息变成电信号进行测量。在测磁仪器中“探头”或“取样装置”就是磁场传感器。随着信息产业、工业自动化、交通运输、电力电子技术、办公自动化、家用电器、医疗仪器等的飞速发

展和电子计算机应用的普及，需用大量的传感器将需进行测量和控制的非电参量转换成可与计算机兼容的信号，作为其输入信号，这就给磁场传感器的快速发展提供了机会，形成了相当可观的磁场传感器产业。本实例中演示了在 Android 设备中使用磁场传感器的方法。

14.4.2 具体实现

本实例的实现文件是 cichangLI.java，在此文件中定义了监听器类对象和注册监听的方法，主要代码如下所示。

```
public class cichangLI extends Activity {
    TextView myTextView1;                                                      //x 方向磁场分量
    TextView myTextView2;                                                      //y 方向磁场分量
    TextView myTextView3;                                                      //z 方向磁场分量
    //SensorManager mySensorManager;                                             //引用 SensorManager 对象
    SensorManagerSimulator mySensorManager;//声明 SensorManagerSimulator 对象，调试时用
    @Override public void onCreate(Bundle savedInstanceState) {               //重写 onCreate()方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                                         //当前的用户界面
        myTextView1 = (TextView) findViewById(R.id.myTextView1);               //得到 myTextView1 引用
        myTextView2 = (TextView) findViewById(R.id.myTextView2);               //得到 myTextView2 引用
        myTextView3 = (TextView) findViewById(R.id.myTextView3);               //得到 myTextView3 引用
        //调试时用
        mySensorManager = SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
        mySensorManager.connectSimulator();                                    //连接 Simulator 服务器
    }
    @SuppressWarnings("deprecation")
     private SensorListener mySensorListener = new SensorListener(){
     @Override
     public void onAccuracyChanged(int sensor, int accuracy) { }               //重写 onAccuracyChanged()方法
     @Override
     public void onSensorChanged(int sensor, float[ ] values) {               //重写 onSensorChanged()方法
          if(sensor == SensorManager.SENSOR_MAGNETIC_FIELD){/ //检查磁场的变化
               myTextView1.setText("x 方向的磁场分量为："+values[0]);//数据显示在 TextView
               myTextView2.setText("y 方向的磁场分量为："+values[1]);//数据显示在 TextView
               myTextView3.setText("z 方向的磁场分量为："+values[2]);//数据显示在 TextView
          }
     }
    };
    @Override
    protected void onResume() {                                                //重写的 onResume()方法
     mySensorManager.registerListener(                                        //注册监听
               mySensorListener,                                              //监听器 SensorListener 对象
               SensorManager.SENSOR_MAGNETIC_FIELD,                           //传感器的类型为加速度
               SensorManager.SENSOR_DELAY_UI                                  //传感器事件传递的频度
               );
     super.onResume();
    }
    @Override
    protected void onPause() {                                                 //重写 onPause()方法
    //取消注册监听器
```

```
        mySensorManager.unregisterListener((SensorEventListener) mySensorListener);
        super.onPause();
    }
}
```

本实例是根据 SensorSimulator 中附带的开源代码改编的，所以在此不再进行详细介绍，读者只需阅读本书附带光盘中的源码即可。

14.5　实现仿微信“摇一摇”效果

实例 161	实现仿微信“摇一摇”效果
源码路径	光盘:\daima\161
视频路径	光盘:\视频\161
实例必备	161.Android 系统中的磁场传感器.pdf

14.5.1　实例说明

传统意义上的加速度传感器是一种能够测量加速力的电子设备。加速力是指当物体在加速过程中作用在物体上的力，就好比地球引力，也就是重力。加速力既可以是个常量，也可以是变量。加速度计有两种，一种是角加速度计，是由陀螺仪（角速度传感器）改进的，另一种是线加速度计。本实例的功能是在界面中实现仿微信“摇一摇”效果。

14.5.2　具体实现

本实例的布局文件是 shake.xml，在界面上方设置了一个图片，在下方设置了按钮控件和文本控件，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_centerInParent="true" >

        <ImageView
            android:id="@+id/shakeBg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@drawable/shake_all" />

        <LinearLayout
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:orientation="vertical" >

          <RelativeLayout
               android:id="@+id/shakeImgUp"
            android:layout_width="fill_parent"
            android:layout_height="190dp"
            android:background="#111111">
               <ImageView
                   android:layout_width="wrap_content"
                   android:layout_height="wrap_content"
                   android:layout_alignParentBottom="true"
                   android:layout_centerHorizontal="true"
                   android:src="@drawable/shake_up"
                    />
          </RelativeLayout>
          <RelativeLayout
               android:id="@+id/shakeImgDown"
            android:layout_width="fill_parent"
            android:layout_height="190dp"
            android:background="#111111">
               <ImageView
                   android:layout_width="wrap_content"
                   android:layout_height="wrap_content"
                   android:layout_centerHorizontal="true"
                   android:src="@drawable/shake_down"
                    />
          </RelativeLayout>
      </LinearLayout>
</RelativeLayout>

<RelativeLayout
      android:id="@+id/shake_title_bar"
        android:layout_width="fill_parent"
        android:layout_height="45dp"
        android:background="@drawable/title_bar"
        android:gravity="center_vertical"   >
            <Button
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="返回"
            android:textSize="14sp"
            android:textColor="#fff"
            android:onClick="shake_activity_back"
                   android:background="@drawable/title_btn_back"/>
            <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
                    android:text="摇一摇"
                    android:layout_centerInParent="true"
                    android:textSize="20sp"
                        android:textColor="#ffffff" />
                    <ImageButton
                        android:layout_width="67dp"
                    android:layout_height="wrap_content"
                    android:layout_alignParentRight="true"
                    android:layout_centerVertical="true"
                    android:layout_marginRight="5dp"
                    android:src="@drawable/mm_title_btn_menu"
                        android:background="@drawable/title_btn_right"
                        android:onClick="linshi"
                        />
        </RelativeLayout>


        <SlidingDrawer
            android:id="@+id/slidingDrawer1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:content="@+id/content"
            android:handle="@+id/handle" >
            <Button
                android:id="@+id/handle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

                android:background="@drawable/shake_report_dragger_up" />
            <LinearLayout
                android:id="@+id/content"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="#f9f9f9" >
                <ImageView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:scaleType="fitXY"
                    android:src="@drawable/shake_line_up" />
            </LinearLayout>
        </SlidingDrawer>
</LinearLayout>
```

程序文件 shakeActivity.java 实现了主 Activity，核心功能是监听设备的摇动方向，定义了摇一摇动画，并设置摇动过程中的震动效果。文件 shakeActivity.java 的主要实现代码如下所示。

```
public class shakeActivity extends Activity{

        ShakeListener mShakeListener = null;
        Vibrator mVibrator;
        private RelativeLayout mImgUp;
        private RelativeLayout mImgDn;
```

```
    private RelativeLayout mTitle;

    private SlidingDrawer mDrawer;
    private Button mDrawerBtn;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.shake);
        //drawerSet ();                    //设置 drawer 监听切换按钮的方向

        mVibrator = (Vibrator)getApplication().getSystemService(VIBRATOR_SERVICE);

        mImgUp = (RelativeLayout) findViewById(R.id.shakeImgUp);
        mImgDn = (RelativeLayout) findViewById(R.id.shakeImgDown);
        mTitle = (RelativeLayout) findViewById(R.id.shake_title_bar);

        mDrawer = (SlidingDrawer) findViewById(R.id.slidingDrawer1);
        mDrawerBtn = (Button) findViewById(R.id.handle);
        mDrawer.setOnDrawerOpenListener(new OnDrawerOpenListener()
        {   public void onDrawerOpened()
            {
                mDrawerBtn.setBackgroundDrawable(getResources().getDrawable(R.drawable.shake_down));
                TranslateAnimation titleup = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-1.0f);
                titleup.setDuration(200);
                titleup.setFillAfter(true);
                mTitle.startAnimation(titleup);
            }
        });
         /*设定 SlidingDrawer 被关闭的事件处理*/
        mDrawer.setOnDrawerCloseListener(new OnDrawerCloseListener()
        {   public void onDrawerClosed()
            {
                mDrawerBtn.setBackgroundDrawable(getResources().getDrawable(R.drawable.shake_
report_dragger_up));
                TranslateAnimation titledn = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-1.0f,Animation.RELATIVE_TO_SELF,0f);
                titledn.setDuration(200);
                titledn.setFillAfter(false);
                mTitle.startAnimation(titledn);
            }
        });

        mShakeListener = new ShakeListener(this);
        mShakeListener.setOnShakeListener(new OnShakeListener() {
            public void onShake() {
                //Toast.makeText(getApplicationContext(), "抱歉，暂时没有找到在同一时刻摇一摇的人。\n
再试一次吧！", Toast.LENGTH_SHORT).show();
                startAnim();   //开始摇一摇手掌动画
```

```
                        mShakeListener.stop();
                        startVibrato();                              //开始震动
                        new Handler().postDelayed(new Runnable(){
                              @Override
                              public void run(){
                                    //Toast.makeText(getApplicationContext(), "抱歉，暂时没有找到\n 在同一时刻摇
一摇的人。\n 再试一次吧！", 500).setGravity(Gravity.CENTER,0,0).show();
                                    Toast mtoast;
                                    mtoast = Toast.makeText(getApplicationContext(),
                                                "抱歉，暂时没有找到\n 在同一时刻摇一摇的人。\n 再试一次吧！", 10);
                                          //mtoast.setGravity(Gravity.CENTER, 0, 0);
                                          mtoast.show();
                                          mVibrator.cancel();
                                          mShakeListener.start();
                              }
                        }, 2000);
                  }
            });
    }
      public void startAnim () {                                      //定义摇一摇动画
            AnimationSet animup = new AnimationSet(true);
            TranslateAnimation mytranslateanimup0 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,
0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-0.5f);
            mytranslateanimup0.setDuration(1000);
            TranslateAnimation mytranslateanimup1 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,
0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,+0.5f);
            mytranslateanimup1.setDuration(1000);
            mytranslateanimup1.setStartOffset(1000);
            animup.addAnimation(mytranslateanimup0);
            animup.addAnimation(mytranslateanimup1);
            mImgUp.startAnimation(animup);

            AnimationSet animdn = new AnimationSet(true);
            TranslateAnimation mytranslateanimdn0 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,
0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,+0.5f);
            mytranslateanimdn0.setDuration(1000);
            TranslateAnimation mytranslateanimdn1 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,
0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-0.5f);
            mytranslateanimdn1.setDuration(1000);
            mytranslateanimdn1.setStartOffset(1000);
            animdn.addAnimation(mytranslateanimdn0);
            animdn.addAnimation(mytranslateanimdn1);
            mImgDn.startAnimation(animdn);
      }
      public void startVibrato(){                                    //定义震动
            mVibrator.vibrate( new long[ ]{500,200,500,200}, -1);//第一个参数是节奏数组，第二个参数是重复次
数，-1 为不重复，非-1 则从 pattern 的指定下标开始重复
      }

      public void shake_activity_back(View v) {                      //标题栏返回按钮
            this.finish();
```

```
    }
    public void linshi(View v) {                                   //标题栏
        startAnim();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mShakeListener != null) {
            mShakeListener.stop();
        }
    }
}
```

上述代码实现了与设备的交互控制——震动，震动是一种提醒或替换铃声的事件，通过上述代码可以了解到如何触发手机震动事件，虽然震动是手机默认的模式，但通过程序的辅助，可以做更精密的控制，诸如震动周期、持续时间等。在设置震动（Vibration）事件中，必须要知道命令其震动的时间长短、震动事件的周期等。而在 Android 中设置的数值都是以毫秒（1000 毫秒=1 秒）来做计算，所以在设置时需要注意，如果设置的时间值太小，会感觉不出来。要想让设备震动，需创建 Vibrator 对象，通过调用 vibrate()方法来达到震动的目的，在 Vibrator 的构造器中有 4 个参数，前 3 个的值是设置震动的大小，此处可以把数值改成一大一小，这样就可以明显感觉出震动的差异，而最后一个值是设置震动的时间。

程序文件 ShakeListener.java 的功能是为设备实现一个设备摇晃的监听器，这一功能是通过传感器实现的。文件 ShakeListener.java 的主要实现代码如下所示。

```
public class ShakeListener implements SensorEventListener {
    //速度阈值，当摇晃速度达到该值后产生作用
    private static final int SPEED_SHRESHOLD = 3000;
    //两次检测的时间间隔
    private static final int UPTATE_INTERVAL_TIME = 70;
    //传感器管理器
    private SensorManager sensorManager;
    //传感器
    private Sensor sensor;
    //重力感应监听器
    private OnShakeListener onShakeListener;
    //上下文
    private Context mContext;
    //手机上一个位置时重力感应坐标
    private float lastX;
    private float lastY;
    private float lastZ;
    //上次检测时间
    private long lastUpdateTime;

    //构造器
    public ShakeListener(Context c) {
        //获得监听对象
        mContext = c;
        start();
    }
```

```
//开始
public void start() {
    //获得传感器管理器
    sensorManager = (SensorManager) mContext
            .getSystemService(Context.SENSOR_SERVICE);
    if (sensorManager != null) {
        //获得重力传感器
        sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }
    //注册
    if (sensor != null) {
        sensorManager.registerListener(this, sensor,
                SensorManager.SENSOR_DELAY_GAME);
    }

}

//停止检测
public void stop() {
    sensorManager.unregisterListener(this);
}

//设置重力感应监听器
public void setOnShakeListener(OnShakeListener listener) {
    onShakeListener = listener;
}

//重力感应器感应获得变化数据
public void onSensorChanged(SensorEvent event) {
    //现在检测时间
    long currentUpdateTime = System.currentTimeMillis();
    //两次检测的时间间隔
    long timeInterval = currentUpdateTime - lastUpdateTime;
    //判断是否达到了检测时间间隔
    if (timeInterval < UPTATE_INTERVAL_TIME)
        return;
    //现在的时间变成 last 时间
    lastUpdateTime = currentUpdateTime;

    //获得 x，y，z 坐标
    float x = event.values[0];
    float y = event.values[1];
    float z = event.values[2];

    //获得 x，y，z 的变化值
    float deltaX = x - lastX;
    float deltaY = y - lastY;
    float deltaZ = z - lastZ;

    //将现在的坐标变成 last 坐标
    lastX = x;
```

```
            lastY = y;
            lastZ = z;

            double speed = Math.sqrt(deltaX * deltaX + deltaY * deltaY + deltaZ
                        * deltaZ)
                        / timeInterval * 10000;
            Log.v("thelog", "===========log==================");
            //达到速度阈值，发出提示
            if (speed >= SPEED_SHRESHOLD) {
                onShakeListener.onShake();
            }
        }

        public void onAccuracyChanged(Sensor sensor, int accuracy) {

        }

        //摇晃监听接口
        public interface OnShakeListener {
            public void onShake();
        }

}
```

在真机中执行后将会实现和微信“摇一摇”类似的效果，如图 14-3 所示。

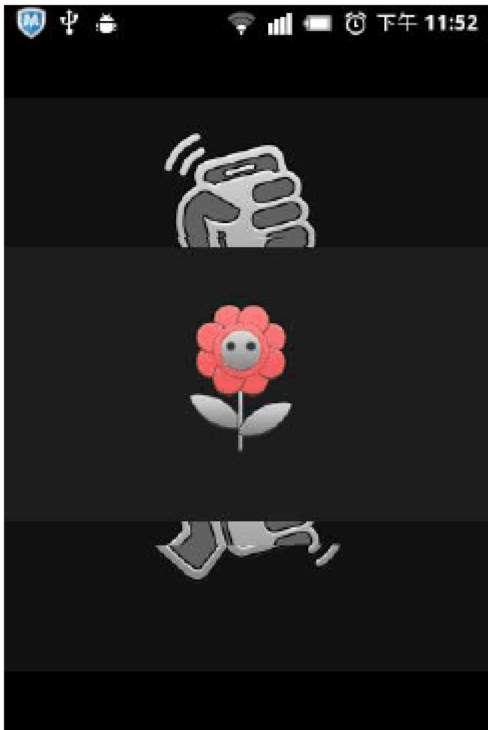


图 14-3 “摇一摇”效果

14.6 测试小球的运动

实例 162	测试小球的运动
源码路径	光盘:\daima\162
视频路径	光盘:\视频\162
实例必备	162.加速度传感器基础.pdf ① 加速度传感器的分类 ② 加速度传感器的主要应用领域

14.6.1　实例说明

线性加速度传感器是加速度传感器的一种，单独介绍的原因是为了和陀螺仪传感器分开。陀螺仪传感器用于测角速度，加速度传感器用于测线性加速度。其中前者利用了惯性原理，而后者利用了力平衡原理。在本实例中，演示了在 Android 设备中使用线性加速度传感器的基本知识。

14.6.2　具体实现

（1）编写布局文件 main.xml，在界面上方设置了一个背景图片，并用文本框显示当前小球在 x 轴、y 轴和 z 轴的重力值。文件 main.xml 的具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+android:id/ball_top"
        android:layout_width="fill_parent"
        android:layout_height="56dp"
        android:orientation="vertical" >

    <TextView
        android:id="@+android:id/ball_prompt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Sensor: 0, 0, 0" />

    <TextView
        android:id="@+id/tv3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginRight="32dp"
        android:text="上下"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="29dp"
        android:text="左右"
        android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
    <TextView
        android:id="@+id/tv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="前后"
        android:textAppearance="?android:attr/textAppearanceLarge" />
     </RelativeLayout>
    <RelativeLayout
        android:id="@+android:id/ball_container"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/bg"
        android:orientation="vertical" >
        <cn.accelerometer.view.BallView
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+android:id/ball"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:scaleType="center"
            android:src="@drawable/ball" />
    </RelativeLayout>
</LinearLayout>
```

（2）编写文件 BallView.java 实现小球视图，具体实现代码如下所示。

```
public class BallView extends ImageView {

     public BallView(Context context) {
          super(context);
     }

    public BallView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public BallView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public void moveTo(int x, int y) {//加一个 TextView，球就不能移动了
        super.setFrame(x, y, x + getWidth(), y + getHeight());//绘制视图，由左上角与右下角确定视图矩形位置
    }
}
```

（3）编写文件 AccelerometerActivity.java，功能是监听传感器的运动轨迹，获取小球在 x 轴、y 轴和 z 轴的重力值。文件 AccelerometerActivity.java 的具体实现代码如下所示。

```
public class AccelerometerActivity extends Activity {
     private static final float MAX_ACCELEROMETER = 9.81f;
     private SensorManager sensorManager;
```

```
    private BallView ball;
    private boolean success = false;
    private boolean init = false;
    private int container_width = 0;
    private int container_height = 0;
    private int ball_width = 0;
    private int ball_height = 0;
    private TextView prompt;
    private TextView tv1;
    private TextView tv2;
    private TextView tv3;
    /*
    private int a=0;
    private int b=0;
     */
   @Override
   public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取感应器管理器
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        prompt = (TextView) findViewById(R.id.ball_prompt);
        tv1 = (TextView)findViewById(R.id.tv1);
        tv2 = (TextView)findViewById(R.id.tv2);
        tv3 = (TextView)findViewById(R.id.tv3);
   }

    @Override
    public void onWindowFocusChanged(boolean hasFocus) {//ball_container 控件显示出来后才能获取其宽和
高，所以用此方法得到其宽和高
          super.onWindowFocusChanged(hasFocus);
          if(hasFocus && !init){
                View container = findViewById(R.id.ball_container);
               container_width = container.getWidth();
               container_height = container.getHeight();
               ball = (BallView) findViewById(R.id.ball);
               ball_width = ball.getWidth();
               ball_height = ball.getHeight();
               moveTo(0f, 0f);
               init = true;
          }
    }

    @Override
    protected void onResume() {
          Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);//获取重力加
速度感应器
          success = sensorManager.registerListener(listener, sensor, SensorManager.SENSOR_DELAY_GAME);
          //注册 listener，第三个参数是检测的精确度
          super.onResume();
    }
```

```
    @Override
     protected void onPause() {
            if(success) sensorManager.unregisterListener(listener);
            super.onPause();
     }

     private SensorEventListener listener = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {



                   /*
                   Button bt1= (Button)findViewById(R.id.bt1);      //测试图片移动
                   bt1.setOnClickListener(new OnClickListener() {
                          @Override
                          public void onClick(View paramView) {
                                 moveTo(0,++b);
                                 //if(b>50)
                                 //      b=0;
                          }
                   });

                   Button bt2= (Button)findViewById(R.id.bt2);
                   bt2.setOnClickListener(new OnClickListener() {
                          @Override
                          public void onClick(View paramView) {
                                 moveTo(0,--b);
                                 //if(b>50)
                                 //      b=0;
                          }
                   });
                   */

                   if (!init) return ;
                   float x = event.values[SensorManager.DATA_X];
            float y = event.values[SensorManager.DATA_Y];
            float z = event.values[SensorManager.DATA_Z];
            prompt.setText("X=" + x + ",Y= " + y + ", Z=" + z);
            //当重力 x、y 为 0 时，球处于中心位置，以 y 为轴心（固定不动）转动手机，x 会在 0～9.81 之间变化，
负号代表方向
            moveTo(-x, y);                                               //x 方向取反


            if(x>0){
                   tv1.setTextColor(Color.WHITE);
                   tv1.setText("向左");
            }

            if(x<0){
                   tv1.setTextColor(Color.CYAN);
```

```
                tv1.setText("向右");
            }

            if(y>0){
                tv2.setTextColor(Color.WHITE);
                tv2.setText("向后");
            }

            if(y<0){
                tv2.setTextColor(Color.RED);
                tv2.setText("向前");
            }

            if(z>0){
                tv3.setTextColor(Color.WHITE);
                tv3.setText("向上");
            }

            if(z<0){
                tv3.setTextColor(Color.YELLOW);
                tv3.setText("向下");
            }


            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }
        };

        private void moveTo(float x, float y) {


            int max_x = (container_width - ball_width) / 2;            //在 x 轴可移动的最大值
            int max_y = (container_height - ball_height) / 2;          //在 y 轴可移动的最大值
            //手机沿 x、y 轴垂直摆放时，自由落体加速度最大为 9.81，当手机沿 x、y 轴成某个角度摆放时，变量 x
和 y 即为该角度的加速度
            float percentageX = x / MAX_ACCELEROMETER;       //得到当前加速度的比率，如果手机沿 x 轴垂直摆
放，比率为 100%，即球在 x 轴上移动到最大值
            float percentageY = y / MAX_ACCELEROMETER;


            int pixel_x = (int) (max_x * percentageX);                 //得到 x 轴偏移量
            int pixel_y = (int) (max_y * percentageY);                 //得到 y 轴偏移量
            //以球在中心位置的坐标为参考点，加上偏移量，得到球的对应位置，然后移动球到该位置

            int x3=max_x + pixel_x;                                    //屏幕中心位置+x 轴偏移
            int y3=max_y + pixel_y;                                    //屏幕中心位置+y 轴偏移

            ball.moveTo(x3, y3);

        }
}
```

执行后会在屏幕上方显示当前滚动小球在 x 轴、y 轴和 z 轴的重力值，如图 14-4 所示。



图 14-4　执行效果

14.7　测试当前设备的 3 个方向值

实例 163	测试当前设备的 3 个方向值
源码路径	光盘:\daima\163
视频路径	光盘:\视频\163
实例必备	163.Android 系统中的加速度传感器.pdf

14.7.1　实例说明

在现实世界中，方向传感器通过对力敏感的传感器，感受手机等设备在变换角度时的重心变化，使手机等设备光标变化位置从而实现选择的功能。方向传感器运用了欧拉角的知识，欧拉角的基本思想是将角位移分解为绕 3 个互相垂直轴的 3 个旋转组成的序列。其实，任意 3 个轴和任意顺序都可以，但最有意义的是使用笛卡儿坐标系并按一定的顺序所组成的旋转序列。本实例演示了在 Android 中使用方向传感器的方法。

14.7.2　具体实现

1．实现布局文件

编写布局文件 main.xml，主要代码如下所示。

```
<TextView
    android:id="@+id/title"
    android:gravity="center_horizontal"
    android:textSize="20px"
```

```
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/title"/><!-- 添加一个 TextView 控件 -->
        <TextView
             android:id="@+id/myTextView1"
             android:textSize="18px"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/myTextView1"/><!-- 添加一个 TextView 控件 -->
        <TextView
             android:id="@+id/myTextView2"
             android:textSize="18px"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/myTextView2"/><!-- 添加一个 TextView 控件 -->
        <TextView
             android:id="@+id/myTextView3"
             android:textSize="18px"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/myTextView3"/><!-- 添加一个 TextView 控件 -->
```

2．实现主程序文件

编写主程序文件 zitaiLI.java，此文件的具体实现流程如下所示。

（1）声明 3 个分别用来显示 Yaw、Pitch、Roll 的 TextView 的引用。

（2）声明 SensorManager 引用，因调试原因用 SensorManagerSimulator 代替。

（3）重写 Activity 的 onCreate()方法，在方法中设置当前的用户界面，然后得到各个控件的引用，并初始化 SensorManager 或 SensorManagerSimulator。

（4）初始化监听器类的对象，在重写的 onSensorChanged()方法中只对姿态 SENSOR_ORIENTATION 变化进行处理，将 3 个姿态值显示到 TextView 中。

（5）重写 Activity 的 onResume()方法，在方法中为 SensorManager 注册监听，此处传入的传感器类型为 SENSOR_ORIENTATION，表示只读取姿态数据。

文件 zitaiLI.java 的主要代码如下所示。

```
public class zitaiLI extends Activity {
    TextView myTextView1;
    TextView myTextView2;
    TextView myTextView3;
    //声明 SensorManagerSimulator 对象，调试时用
    SensorManagerSimulator mySensorManager;
    @Override
    public void onCreate(Bundle savedInstanceState) {                    //重写 onCreate()方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                                   //设置用户界面
        myTextView1 = (TextView) findViewById(R.id.myTextView1);     //myTextView1 的引用
        myTextView2 = (TextView) findViewById(R.id.myTextView2);     //myTextView2 的引用
        myTextView3 = (TextView) findViewById(R.id.myTextView3);     //myTextView3 的引用
        //mySensorManager =
        //(SensorManager)getSystemService(SENSOR_SERVICE);              //获得 SensorManager
```

```
        //调试时用
        mySensorManager = SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
        mySensorManager.connectSimulator();                    //与 Simulator 服务器连接
    }
    private SensorListener mySensorListener = new SensorListener(){
     @Override
     public void onAccuracyChanged(int sensor, int accuracy) {}        //重写 onAccuracyChanged()方法
     @Override
     public void onSensorChanged(int sensor, float[ ] values) {        //重写 onSensorChanged()方法
            if(sensor == SensorManager.SENSOR_ORIENTATION){            //检查姿态变化
                myTextView1.setText("Yaw 为："+values[0]);            //TextView 数据显示
                myTextView2.setText("Pitch 为："+values[1]);          //TextView 数据显示
                myTextView3.setText("Roll 为："+values[2]);           //TextView 数据显示
            }
     }
    };
    @Override
    protected void onResume() {                                        //重写的 onResume()方法
     mySensorManager.registerListener(                                 //注册监听
                mySensorListener,                                      //监听器 SensorListener 对象
                SensorManager.SENSOR_ORIENTATION,                      //姿态传感器的类型
                SensorManager.SENSOR_DELAY_UI                          //传感器事件传递频度
                );
     super.onResume();
    }
    @Override
    protected void onPause() {                                         //重写 onPause()方法
     mySensorManager.unregisterListener(mySensorListener);             //取消注册监听器
     super.onPause();
    }
}
```

本实例是根据 SensorSimulator 中附带的开源代码改编的，所以在此不再进行详细介绍，读者只需阅读本书附带光盘中的源码即可。

14.8　确定设备当前的具体方向

实例 164	测试当前设备的 3 个方向值
源码路径	https://github.com/gast-lib/gast-lib
视频路径	光盘:\视频\164
实例必备	164.方向传感器基础.pdf ① 方向传感器必备知识 ② Android 中的方向传感器

14.8.1　实例说明

旋转向量传感器也被称为旋转矢量传感器，简称 RV-sensor。旋转矢量代表设备的方向，是一个将

坐标轴和角度混合计算得到的数据。本实例联合使用了旋转向量传感器、磁场传感器、重力传感器和加速度传感器，功能是获取当前设备的方向。

14.8.2　具体实现

本实例源码是开源代码，来源于如下地址，读者可以自行登录并下载。

```
https://github.com/gast-lib/gast-lib/
```

1．实现主 Activity

本实例的主 Activity 是 DetermineOrientationActivity，通过布局文件 determine_orientation.xml 实现布局，在屏幕中提供一组单选按钮供用户选择所需要的传感器，并在屏幕下方显示具体的显示传感器返回的数据。布局文件 determine_orientation.xml 的具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <RadioGroup android:id="@+id/sensorSelector"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true" >

        <RadioButton android:id="@+id/gravitySensor"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/gravitySensorLabel"
            android:checked="true"
            android:onClick="onSensorSelectorClick" />

        <RadioButton android:id="@+id/accelerometerMagnetometer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/accelerometerMagnetometerLabel"
            android:checked="false"
            android:onClick="onSensorSelectorClick" />

        <RadioButton android:id="@+id/gravityMagnetometer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/gravityMagnetometerLabel"
            android:checked="false"
            android:onClick="onSensorSelectorClick" />

        <RadioButton android:id="@+id/rotationVector"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/rotationVectorLabel"
            android:checked="false"
            android:onClick="onSensorSelectorClick" />
```

```
</RadioGroup>

<ToggleButton android:id="@+id/ttsNotificationsToggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/speakOrientationLabel"
    android:checked="true"
    android:layout_below="@id/sensorSelector"
    android:textOn="@string/ttsNotificationsOn"
    android:textOff="@string/ttsNotificationsOff"
    android:onClick="onTtsNotificationsToggleButtonClicked" />

<TextView android:id="@+id/selectedSensorLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/selectedSensorLabel"
    android:layout_below="@id/ttsNotificationsToggleButton"
    android:layout_marginRight="5dip" />

<TextView android:id="@+id/selectedSensorValue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/selectedSensorLabel"
    android:layout_alignTop="@id/selectedSensorLabel"
    android:layout_alignBottom="@id/selectedSensorLabel" />

<TextView android:id="@+id/orientationLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/orientationLabel"
    android:layout_below="@id/selectedSensorValue"
    android:layout_marginRight="5dip" />

<TextView android:id="@+id/orientationValue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/orientationLabel"
    android:layout_alignTop="@id/orientationLabel"
    android:layout_alignBottom="@id/orientationLabel" />

<TextView android:id="@+id/sensorXLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/orientationValue"
    android:layout_marginRight="5dip" />

<TextView android:id="@+id/sensorXValue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/sensorXLabel"
    android:layout_alignTop="@id/sensorXLabel"
    android:layout_alignBottom="@id/sensorXLabel" />
```

```
    <TextView android:id="@+id/sensorYLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/sensorXLabel"
        android:layout_marginRight="5dip" />

    <TextView android:id="@+id/sensorYValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/sensorYLabel"
        android:layout_alignTop="@id/sensorYLabel"
        android:layout_alignBottom="@id/sensorYLabel" />

    <TextView android:id="@+id/sensorZLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/sensorYLabel"
        android:layout_marginRight="5dip" />

    <TextView android:id="@+id/sensorZValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/sensorZLabel"
        android:layout_alignTop="@id/sensorZLabel"
        android:layout_alignBottom="@id/sensorZLabel" />
</RelativeLayout>
```

主 Activity 程序文件 DetermineOrientationActivity.java 的功能是获取 SensorManager 的引用，根据用户单选按钮的选择来注册这个传感器，然后调用这个传感器来获取数据。假如选择的是重力传感器，则会注册重力传感器，然后获取数组 SensorEvent.Values 中的 x、y 和 z 轴上的重力大小。当选择使用旋转向量传感器时，会调用方法 determineOrientation(rotationMatrix)根据给出的旋转矩阵计算具体的方向。当使用者确定了设备的具体朝向时，会使用文本转语音的功能提醒用户。本实例的语音提醒功能是通过 TTS 机制实现的。文件 DetermineOrientationActivity.java 的具体实现代码如下所示。

```
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        super.setContentView(R.layout.determine_orientation);

        // Keep the screen on so that changes in orientation can be easily
        // observed
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        // Set up stream to use for Text-To-Speech
        ttsParams = new HashMap<String, String>();
        ttsParams.put(Engine.KEY_PARAM_STREAM, String.valueOf(TTS_STREAM));

        // Set the volume control to use the same stream as TTS which allows
        // the user to easily adjust the TTS volume
        this.setVolumeControlStream(TTS_STREAM);
```

```
        // Get a reference to the sensor service
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        // Initialize references to the UI views that will be updated in the
        // code
        sensorSelector = (RadioGroup) findViewById(R.id.sensorSelector);
        selectedSensorValue = (TextView) findViewById(R.id.selectedSensorValue);
        orientationValue = (TextView) findViewById(R.id.orientationValue);
        sensorXLabel = (TextView) findViewById(R.id.sensorXLabel);
        sensorXValue = (TextView) findViewById(R.id.sensorXValue);
        sensorYLabel = (TextView) findViewById(R.id.sensorYLabel);
        sensorYValue = (TextView) findViewById(R.id.sensorYValue);
        sensorZLabel = (TextView) findViewById(R.id.sensorZLabel);
        sensorZValue = (TextView) findViewById(R.id.sensorZValue);
        ttsNotificationsToggleButton =
                (ToggleButton) findViewById(R.id.ttsNotificationsToggleButton);

        // Retrieve stored preferences
        preferences = getPreferences(MODE_PRIVATE);
        ttsNotifications =
                preferences.getBoolean(TTS_NOTIFICATION_PREFERENCES_KEY, true);
    }

    @Override
    protected void onResume()
    {
        super.onResume();

        ttsNotificationsToggleButton.setChecked(ttsNotifications);
        updateSelectedSensor();
    }

    @Override
    protected void onPause()
    {
        super.onPause();

        // Unregister updates from sensors
        sensorManager.unregisterListener(this);

        // Shutdown TTS facility
        if (tts != null)
        {
            tts.shutdown();
        }
    }

    @Override
    public void onSensorChanged(SensorEvent event)
    {
        float[ ] rotationMatrix;
```

```
switch (event.sensor.getType())
{
    case Sensor.TYPE_GRAVITY:
        sensorXLabel.setText(R.string.xAxisLabel);
        sensorXValue.setText(String.valueOf(event.values[0]));

        sensorYLabel.setText(R.string.yAxisLabel);
        sensorYValue.setText(String.valueOf(event.values[1]));

        sensorZLabel.setText(R.string.zAxisLabel);
        sensorZValue.setText(String.valueOf(event.values[2]));

        sensorYLabel.setVisibility(View.VISIBLE);
        sensorYValue.setVisibility(View.VISIBLE);
        sensorZLabel.setVisibility(View.VISIBLE);
        sensorZValue.setVisibility(View.VISIBLE);

        if (selectedSensorId == R.id.gravitySensor)
        {
            if (event.values[2] >= GRAVITY_THRESHOLD)
            {
                onFaceUp();
            }
            else if (event.values[2] <= (GRAVITY_THRESHOLD * -1))
            {
                onFaceDown();
            }
        }
        else
        {
            accelerationValues = event.values.clone();
            rotationMatrix = generateRotationMatrix();

            if (rotationMatrix != null)
            {
                determineOrientation(rotationMatrix);
            }
        }

        break;
    case Sensor.TYPE_ACCELEROMETER:
        accelerationValues = event.values.clone();
        rotationMatrix = generateRotationMatrix();

        if (rotationMatrix != null)
        {
            determineOrientation(rotationMatrix);
        }
        break;
    case Sensor.TYPE_MAGNETIC_FIELD:
        magneticValues = event.values.clone();
        rotationMatrix = generateRotationMatrix();
```

```
                    if (rotationMatrix != null)
                    {
                        determineOrientation(rotationMatrix);
                    }
                    break;
                case Sensor.TYPE_ROTATION_VECTOR:

                    rotationMatrix = new float[16];
                    SensorManager.getRotationMatrixFromVector(rotationMatrix,
                            event.values);
                    determineOrientation(rotationMatrix);
                    break;
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy)
        {
            Log.d(TAG,
                    String.format("Accuracy for sensor %s = %d",
                    sensor.getName(), accuracy));
        }

        /**
         * Generates a rotation matrix using the member data stored in
         * accelerationValues and magneticValues.
         *
         * @return The rotation matrix returned from
         * {@link android.hardware.SensorManager#getRotationMatrix(float[ ], float[ ], float[ ], float[ ])}
         * or <code>null</code> if either <code>accelerationValues</code> or
         * <code>magneticValues</code> is null.
         */
        private float[ ] generateRotationMatrix()
        {
            float[ ] rotationMatrix = null;

            if (accelerationValues != null && magneticValues != null)
            {
                rotationMatrix = new float[16];
                boolean rotationMatrixGenerated;
                rotationMatrixGenerated =
                        SensorManager.getRotationMatrix(rotationMatrix,
                        null,
                        accelerationValues,
                        magneticValues);

                if (!rotationMatrixGenerated)
                {
                    Log.w(TAG, getString(R.string.rotationMatrixGenFailureMessage));

                    rotationMatrix = null;
```

```
            }
        }

        return rotationMatrix;
    }

    /**
     * Uses the last read accelerometer and gravity values to determine if the
     * device is face up or face down.
     *
     * @param rotationMatrix The rotation matrix to use if the orientation
     * calculation
     */
    private void determineOrientation(float[ ] rotationMatrix)
    {
        float[ ] orientationValues = new float[3];
        SensorManager.getOrientation(rotationMatrix, orientationValues);

        double azimuth = Math.toDegrees(orientationValues[0]);
        double pitch = Math.toDegrees(orientationValues[1]);
        double roll = Math.toDegrees(orientationValues[2]);

        sensorXLabel.setText(R.string.azimuthLabel);
        sensorXValue.setText(String.valueOf(azimuth));

        sensorYLabel.setText(R.string.pitchLabel);
        sensorYValue.setText(String.valueOf(pitch));

        sensorZLabel.setText(R.string.rollLabel);
        sensorZValue.setText(String.valueOf(roll));

        sensorYLabel.setVisibility(View.VISIBLE);
        sensorYValue.setVisibility(View.VISIBLE);
        sensorZLabel.setVisibility(View.VISIBLE);
        sensorZValue.setVisibility(View.VISIBLE);

        if (pitch <= 10)
        {
            if (Math.abs(roll) >= 170)
            {
                onFaceDown();
            }
            else if (Math.abs(roll) <= 10)
            {
                onFaceUp();
            }
        }
    }

    /**
     * Handler for device being face up.
     */
```

```
private void onFaceUp()
{
    if (!isFaceUp)
    {
        if (tts != null && ttsNotificationsToggleButton.isChecked())
        {
            tts.speak(getString(R.string.faceUpText),
                    TextToSpeech.QUEUE_FLUSH,
                    ttsParams);
        }

        orientationValue.setText(R.string.faceUpText);
        isFaceUp = true;
    }
}

/**
 * Handler for device being face down.
 */
private void onFaceDown()
{
    if (isFaceUp)
    {
        if (tts != null && ttsNotificationsToggleButton.isChecked())
        {
            tts.speak(getString(R.string.faceDownText),
                    TextToSpeech.QUEUE_FLUSH,
                    ttsParams);
        }

        orientationValue.setText(R.string.faceDownText);
        isFaceUp = false;
    }
}

/**
 * Updates the views for when the selected sensor is changed
 */
private void updateSelectedSensor()
{
    // Clear any current registrations
    sensorManager.unregisterListener(this);

    // Determine which radio button is currently selected and enable the
    // appropriate sensors
    selectedSensorId = sensorSelector.getCheckedRadioButtonId();
    if (selectedSensorId == R.id.accelerometerMagnetometer)
    {
        sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                RATE);
```

```
            sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                    RATE);
        }
        else if (selectedSensorId == R.id.gravityMagnetometer)
        {
            sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY),
                    RATE);

            sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                    RATE);
        }
        else if ((selectedSensorId == R.id.gravitySensor))
        {
            sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY),
                    RATE);
        }
        else
        {
            sensorManager.registerListener(this,
                    sensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR),
                    RATE);
        }

        // Update the label with the currently selected sensor
        RadioButton selectedSensorRadioButton =
                (RadioButton) findViewById(selectedSensorId);
        selectedSensorValue.setText(selectedSensorRadioButton.getText());
    }

    /**
     * Handles click event for the sensor selector.
     *
     * @param view The view that was clicked
     */
    public void onSensorSelectorClick(View view)
    {
        updateSelectedSensor();
    }

    /**
     * Handles click event for the TTS toggle button.
     *
     * @param view The view for the toggle button
     */
    public void onTtsNotificationsToggleButtonClicked(View view)
    {
        ttsNotifications = ((ToggleButton) view).isChecked();
```

```
            preferences.edit()
                .putBoolean(TTS_NOTIFICATION_PREFERENCES_KEY, ttsNotifications)
                .commit();
    }

    @Override
    public void onSuccessfulInit(TextToSpeech tts)
    {
        super.onSuccessfulInit(tts);
        this.tts = tts;
    }

    @Override
    protected void receiveWhatWasHeard(List<String> heard, float[ ] confidenceScores)
    {
        // no-op
    }
}
```

2．获取设备的旋转向量

编写文件 NorthFinder.java，首先获取设备的旋转向量，并将旋转向量的坐标映射到摄像头的轴上。如果取消了对方法 remapCoordinateSystem()的调用，则将当前设备指向北方，而并不是将后置摄像头指向北方。除此之外，在此文件中还使用 OpenGL 改变了屏幕的颜色，当后置摄像头指向北方时（允许误差 20°内），将屏幕颜色从红色变为绿色。文件 NorthFinder.java 的具体实现代码如下所示。

```
public class NorthFinder extends Activity implements SensorEventListener
{
    private static final int ANGLE = 20;

    private TextView tv;
    private GLSurfaceView mGLSurfaceView;
    private MyRenderer mRenderer;
    private SensorManager mSensorManager;
    private Sensor mRotVectSensor;
    private float[ ] orientationVals = new float[3];

    private final float[ ] mRotationMatrix = new float[16];

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.sensors_north_main);

        mRenderer = new MyRenderer();
        mGLSurfaceView = (GLSurfaceView) findViewById(R.id.glsurfaceview);
        mGLSurfaceView.setRenderer(mRenderer);

        tv = (TextView) findViewById(R.id.tv);

        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

```
        mRotVectSensor =
                mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);

    }

    @Override
    protected void onResume()
    {
        super.onResume();
        mSensorManager.registerListener(this, mRotVectSensor, 10000);
    }

    @Override
    protected void onPause()
    {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    @Override
    public void onSensorChanged(SensorEvent event)
    {
        // It is good practice to check that we received the proper sensor event
        if (event.sensor.getType() == Sensor.TYPE_ROTATION_VECTOR)
        {
            // Convert the rotation-vector to a 4×4 matrix.
            SensorManager.getRotationMatrixFromVector(mRotationMatrix,
                    event.values);
            SensorManager
                    .remapCoordinateSystem(mRotationMatrix,
                            SensorManager.AXIS_X, SensorManager.AXIS_Z,
                            mRotationMatrix);
            SensorManager.getOrientation(mRotationMatrix, orientationVals);

            // Optionally convert the result from radians to degrees
            orientationVals[0] = (float) Math.toDegrees(orientationVals[0]);
            orientationVals[1] = (float) Math.toDegrees(orientationVals[1]);
            orientationVals[2] = (float) Math.toDegrees(orientationVals[2]);

            tv.setText(" Yaw: " + orientationVals[0] + "\n Pitch: "
                    + orientationVals[1] + "\n Roll (not used): "
                    + orientationVals[2]);

        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy)
    {
        // no-op
    }

    class MyRenderer implements GLSurfaceView.Renderer
```

```
    {
        public void onDrawFrame(GL10 gl)
        {
            // Clear screen
            gl.glClear(GL10.GL_COLOR_BUFFER_BIT);

            // Detect if the device is pointing within +/- ANGLE of north
            if (orientationVals[0] < ANGLE && orientationVals[0] > -ANGLE
                    && orientationVals[1] < ANGLE
                    && orientationVals[1] > -ANGLE)
            {
                gl.glClearColor(0, 1, 0, 1); // Make background green
            }
            else
            {
                gl.glClearColor(1, 0, 0, 1); // Make background red
            }
        }

        @Override
        public void onSurfaceChanged(GL10 gl, int width, int height)
        {
            // no-op
        }

        @Override
        public void onSurfaceCreated(GL10 gl, EGLConfig config)
        {
            // no-op
        }
    }
}
```

至此，整个实例的核心代码介绍完毕。为节省篇幅，其余的代码将不再进行详细讲解。

14.9 使用距离传感器实现自动锁屏功能

实例 165	使用距离传感器实现自动锁屏功能
源码路径	光盘:\daima\165
视频路径	光盘:\视频\165
实例必备	165.距离传感器基础.pdf ① 距离传感器介绍 ② Android 系统中的距离传感器

14.9.1 实例说明

在 Android 设备应用程序开发过程中，经常需要检测设备的运动数据，例如，设备的运动速率和

运动距离等。这些数据对于健身类设备来说，都是十分重要的数据，例如，健身手表可以及时测试晨练的运动距离和速率。在 Android 系统中，通常使用加速度传感器、线性加速度传感器和距离传感器来检测设备的运动数据。在本实例中，详细讲解了在 Android 设备中检测运动数据的基本知识。

14.9.2　具体实现

（1）编写布局文件 activity_main.xml，功能是在屏幕中分别设置“启动服务”“停止服务”“退出”3 个按钮，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/title_tv"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="@string/title" />

     <Button
        android:id="@+id/start"
        android:layout_below="@id/title_tv"
        android:layout_centerHorizontal="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="@string/start" />

     <Button
        android:id="@+id/stop"
        android:layout_below="@id/start"
        android:layout_centerHorizontal="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="@string/stop" />

     <Button
        android:id="@+id/exit"
        android:layout_below="@id/stop"
        android:layout_centerHorizontal="true"
        android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="@string/exit" />

        <TextView
            android:id="@+id/sensortitle_tv"
            android:layout_below="@id/exit"
            android:layout_centerHorizontal="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="@string/sensorinfo" />

         <TextView
            android:id="@+id/sensorinfo_tv"
            android:layout_below="@id/sensortitle_tv"
            android:layout_centerHorizontal="true"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="@string/sensorinfo" />
</RelativeLayout>
```

（2）编写文件 MainActivity.java，在启动时显示传感器名和版本号，并根据用户的按钮操作执行对应的事件处理程序。文件 MainActivity.java 的具体实现代码如下所示。

```
public class MainActivity extends Activity {

    private Button start;
    private Button stop;
    private Button exit;
    private TextView sensorinfo_tv;
    private Intent intent;
    private SensorManager sm = null;
    private Sensor promixty = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (null == sm) {
            sm = (SensorManager) getSystemService(SENSOR_SERVICE);    //获取传感器管理类
            promixty = sm.getDefaultSensor(Sensor.TYPE_PROXIMITY);     //获取距离传感器
        }
        String sensorInfo;
        if (null != promixty) {
            sensorInfo = "传感器名称： " + promixty.getName() + "\n"
                    + "   设备版本： " + promixty.getVersion() + "\n" + "   供应商： "
                    + promixty.getVendor() + "\n";
        }
        else {
```

```
            sensorInfo = "无法获取距离传感器信息，可能是您的手机不支持该传感器。";
        }
        initUI();
        intent = new Intent("org.hq.autoLockService");
        start.setOnClickListener( new OnClickListener() {
            @Override
            public void onClick(View v) {
                //开启服务
                start();
            }
        });
        stop.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //停止服务
                stop();
            }
        });
        exit.setOnClickListener( new OnClickListener() {

            @Override
            public void onClick(View v) {
                //结束本次 Activity
                finish();
            }
        });
        sensorinfo_tv.setText(sensorInfo);
    }

    private void initUI(){
        start = (Button) super.findViewById(R.id.start);
        stop = (Button) super.findViewById(R.id.stop);
        exit = (Button) super.findViewById(R.id.exit);
        sensorinfo_tv = (TextView) super.findViewById(R.id.sensorinfo_tv);
    }

    private void start(){
        Bundle bundle = new Bundle();
        bundle.putInt("distance", 3);
        bundle.putBoolean("activited", true);
        intent.putExtras(bundle);
        startService(intent);                              //startService
        //结束本次 Activity
        //finish();
    }
    //终止服务
    private void stop(){
        stopService(intent);
        Toast.makeText(this, "已停止后台服务。", Toast.LENGTH_SHORT).show();
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

（3）编写文件 AutoLockService.java 实现自动锁屏服务，通过距离传感器监听距离，自动进入锁屏状态。文件 AutoLockService.java 的具体实现代码如下所示。

```
public class AutoLockService extends Service implements SensorEventListener {

    private SensorManager sm = null;
    private Sensor promixty = null;
    //默认启用锁屏
    private static boolean ACTIVITED = true;
    //锁屏距离（单位：厘米）
    private static int LOCK_DIST = 3;

    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        if (null == sm) {
            sm = (SensorManager) getSystemService(SENSOR_SERVICE);      //获取传感器管理类
            promixty = sm.getDefaultSensor(Sensor.TYPE_PROXIMITY);      //获取距离传感器
        }
        //显示距离传感器信息
        if (null != promixty) {
            Toast.makeText(this, "已创建后台服务。", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "无法找到距离传感器", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if( null != sm ){
            //撤销监听器
            sm.unregisterListener(this);
        }
    }

    @Override
```

```
    public void onStart(Intent intent, int startId) {
        if (intent != null) {
            Bundle bundle = intent.getExtras();
            Toast.makeText(this, "后台服务已启动。", Toast.LENGTH_SHORT).show();
            //从 Intent 中获取设置参数
            if (bundle != null) {
                int dist = bundle.getInt("distance");
                ACTIVITED = bundle.getBoolean("activited");
                if (dist > 0 && dist < 9) {
                    LOCK_DIST = dist;
                }
            }
            //注册监听器
            sm.registerListener(this, promixty,SensorManager.SENSOR_DELAY_NORMAL);
        }
    }

    //监听精度变化
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        Toast.makeText(this, "距离传感器 promixty 精度变为" + accuracy,
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.values[0] < LOCK_DIST)                        //距离小于 5，锁屏
        {
            if (ACTIVITED) {
                lockScreen();
            }
        }
    }

    //跳至锁屏页面
    private void lockScreen() {
        Intent intent = new Intent();
        //在 Activity 之外启动，要加上 FLAG_ACTIVITY_NEW_TASK flag
        intent.setFlags( Intent.FLAG_ACTIVITY_NEW_TASK );
        intent.setClass(this, LockScreen.Controller.class);
        startActivity(intent);
    }
}
```

（4）编写文件 LockScreen.java，功能是实现锁屏功能，在锁屏之前需要先获取锁屏权限。文件 LockScreen.java 的具体实现代码如下所示。

```
public class LockScreen extends DeviceAdminReceiver {
    static final int RESULT_ENABLE = 1;

    public static class Controller extends Activity {
```

```
            DevicePolicyManager mDPM;
            ComponentName mDeviceAdminSample;

            @Override
            protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                //首先要获得 Android 设备管理代理
                mDPM = (DevicePolicyManager) getSystemService(Context.DEVICE_POLICY_SERVICE);

                //LockScreen 继承自 DeviceAdminReceiver
                mDeviceAdminSample = new ComponentName(Controller.this,
                        LockScreen.class);
                //得到当前设备管理器有没有激活
                boolean active = mDPM.isAdminActive(mDeviceAdminSample);
                if (!active) {
                    //如果没有激活，则提示用户激活（第一次运行程序时）
                    getAdmin();
                } else {
                    //如果已经激活，则立即锁屏
                    mDPM.lockNow();
                }
                //锁屏之后就立即关掉 Activity，避免资源的浪费
                //android.os.Process.killProcess(android.os.Process.myPid());
                finish();
            }

            //获取锁屏权限
            public void getAdmin() {
                // Launch the activity to have the user enable our admin.
                Intent intent = new Intent(
                        DevicePolicyManager.ACTION_ADD_DEVICE_ADMIN);
                intent.putExtra(DevicePolicyManager.EXTRA_DEVICE_ADMIN,
                        mDeviceAdminSample);
                intent.putExtra(DevicePolicyManager.EXTRA_ADD_EXPLANATION,
                        "欢迎您的使用！在第一次使用时，请授予该程序锁屏权限。");
                startActivityForResult(intent, RESULT_ENABLE);
            }
        }
}
```

（5）在文件 AndroidManifest.xml 中声明权限，特别是需要注册一个广播接收者，具体实现代码如下所示。

```
        <!-- 注册锁屏 Activity -->
        <activity android:name="org.lock.LockScreen$Controller" >
        </activity>

        <service
            android:name="org.lock.AutoLockService"
            android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
            android:enabled="true" >
```

```
            <intent-filter>
                <action android:name="org.lock.autoLockService" />
            </intent-filter>
        </service>
        <receiver
            android:name="org.lock.LockScreen"
            android:permission="android.permission.BIND_DEVICE_ADMIN" >
            <meta-data
                android:name="android.app.device_admin"
                android:resource="@xml/device_admin_sample" />

            <intent-filter>
                <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
            </intent-filter>
        </receiver>
```

至此，整个实例介绍完毕，执行后的效果如图 14-5 所示。



图 14-5　执行效果